Untitled0

April 21, 2024

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as mt
```

[4]:
```
dataset= pd.read_csv("/content/Iris.csv")
```

[5]:
```
dataset.isnull()
```

[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
..	…	…	…	…	…	…
145	False	False	False	False	False	False
146	False	False	False	False	False	False
147	False	False	False	False	False	False
148	False	False	False	False	False	False
149	False	False	False	False	False	False

[150 rows x 6 columns]

[6]:
```
dataset.isnull()
```

[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
..	…	…	…	…	…	…
145	False	False	False	False	False	False
146	False	False	False	False	False	False
147	False	False	False	False	False	False
148	False	False	False	False	False	False
149	False	False	False	False	False	False

```
[150 rows x 6 columns]
```

[7]:
```
dataset.describe()
```

[7]:
```
                Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count   150.000000     150.000000    150.000000     150.000000    150.000000
mean     75.500000       5.843333      3.054000       3.758667      1.198667
std      43.445368       0.828066      0.433594       1.764420      0.763161
min       1.000000       4.300000      2.000000       1.000000      0.100000
25%      38.250000       5.100000      2.800000       1.600000      0.300000
50%      75.500000       5.800000      3.000000       4.350000      1.300000
75%     112.750000       6.400000      3.300000       5.100000      1.800000
max     150.000000       7.900000      4.400000       6.900000      2.500000
```

[8]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

[9]:
```
dataset
```

[9]:
```
       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0       1            5.1           3.5            1.4           0.2
1       2            4.9           3.0            1.4           0.2
2       3            4.7           3.2            1.3           0.2
3       4            4.6           3.1            1.5           0.2
4       5            5.0           3.6            1.4           0.2
..    ...            ...           ...            ...           ...
145   146            6.7           3.0            5.2           2.3
146   147            6.3           2.5            5.0           1.9
147   148            6.5           3.0            5.2           2.0
148   149            6.2           3.4            5.4           2.3
149   150            5.9           3.0            5.1           1.8

            Species
```

```
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

[10]:
```
dataset['Species'].replace(['Iris-setosa','Iris.
↪virginica','Iris-versicolour'],[0,1,2],inplace=True)
```

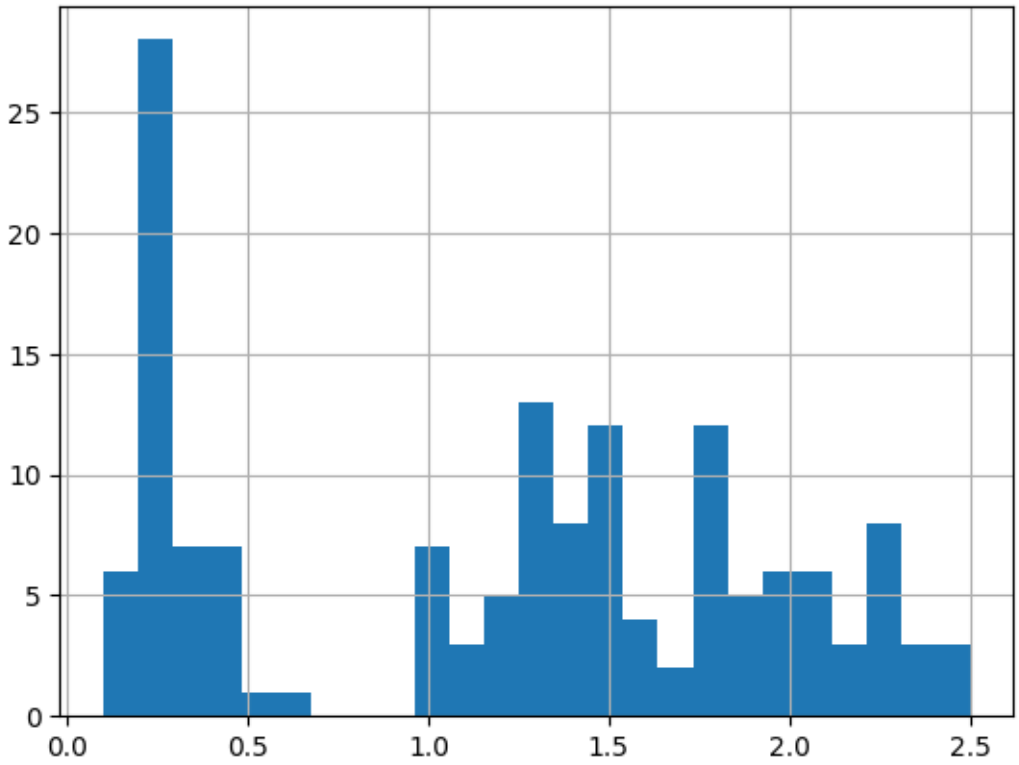
[11]:
```
dataset.head()
```

[11]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0

[12]:
```
dataset['PetalWidthCm'].hist(bins=25)
```

[12]:
```
<Axes: >
```


25
20
15
10
5
0
0.0
0.5
1.0
1.5
2.0
2.5

Practical2

April 21, 2024

[ ]:
```
import pandas as pd
import numpy as nm
```

[ ]:
```
dataset=pd.read_csv('/content/StudentsPerformance.csv')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-24f3b27c6fa1> in <cell line: 1>()
----> 1 dataset=pd.read_csv('/content/StudentsPerformance.csv')

NameError: name 'pd' is not defined
```

[ ]:
```
dataset.head()
```

[ ]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

[ ]:
```
dataset.isnull().sum()
```

[ ]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
```

```
writing score                  0
dtype: int64
```

[ ]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64 
 6   reading score                1000 non-null   int64 
 7   writing score                1000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

[ ]:
```
numeric_variable = dataset.columns[dataset.dtypes !='O']
```

[ ]:
```
numeric_variable
```

[ ]:
```
Index(['math score', 'reading score', 'writing score'], dtype='object')
```

[ ]:
```
categorical_variable = dataset.columns[dataset.dtypes =='O']
```

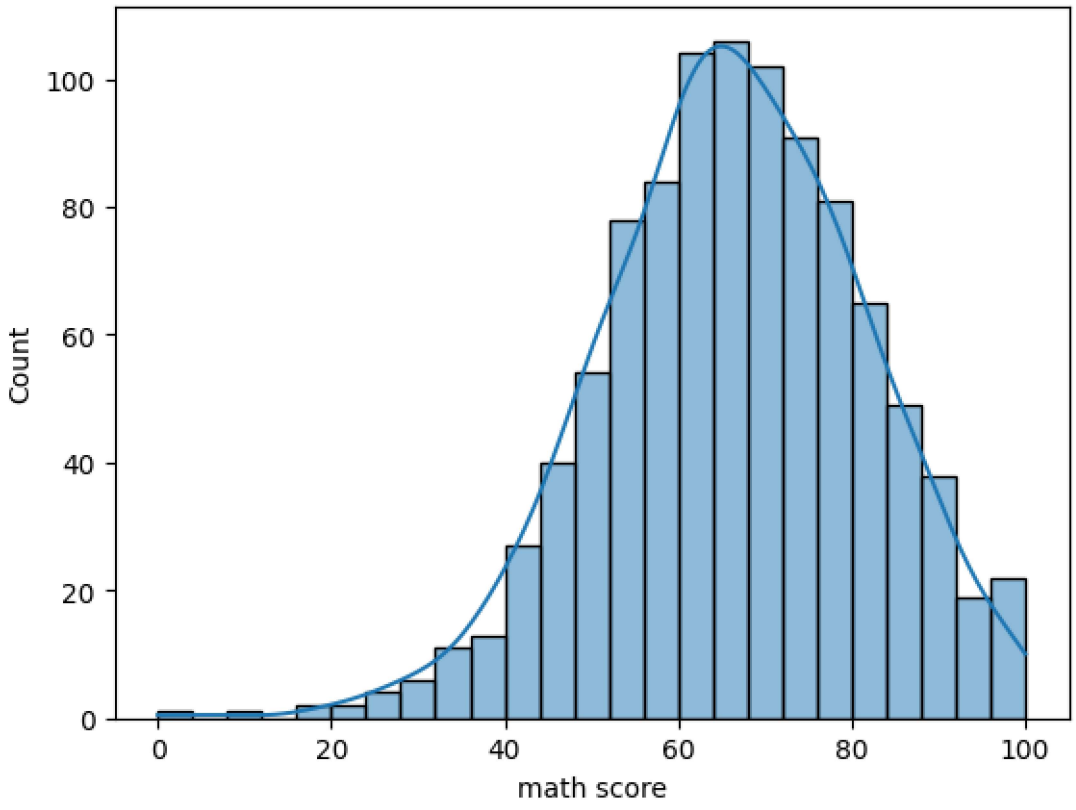
[ ]:
```
categorical_variable
```

[ ]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course'],
      dtype='object')
```

[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[ ]:
```
sns.histplot(data=dataset,x='math score',kde=True)
```

[ ]:
```
<Axes: xlabel='math score', ylabel='Count'>
```

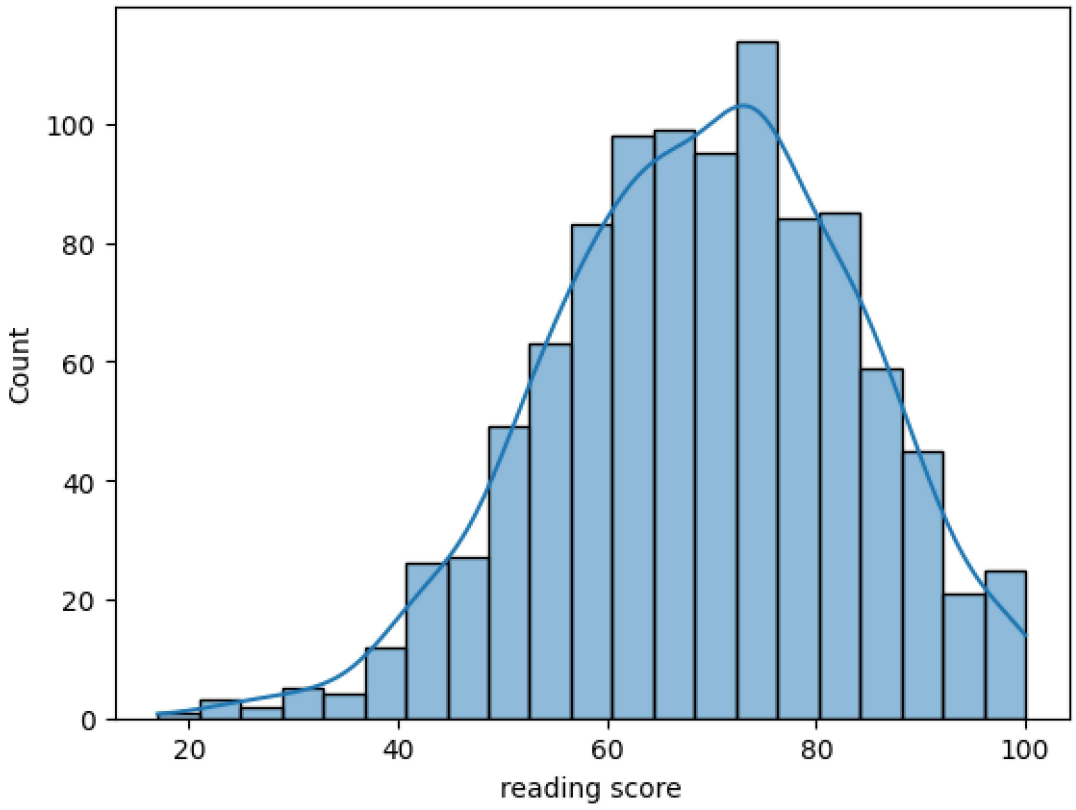


[ ]:

[ ]:

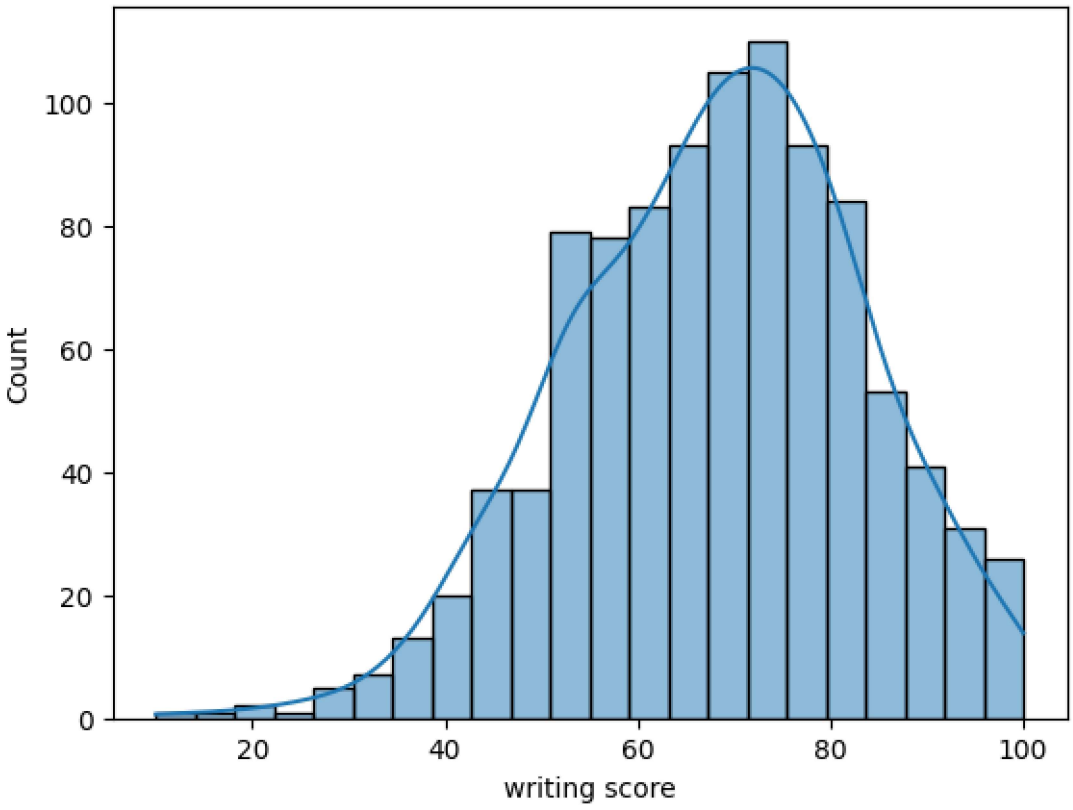
[ ]:
```
sns.histplot(data=dataset,x='reading score',kde=True)
```

[ ]:
```
<Axes: xlabel='reading score', ylabel='Count'>
```



[ ]:
```
sns.histplot(data=dataset,x='writing score',kde=True)
```

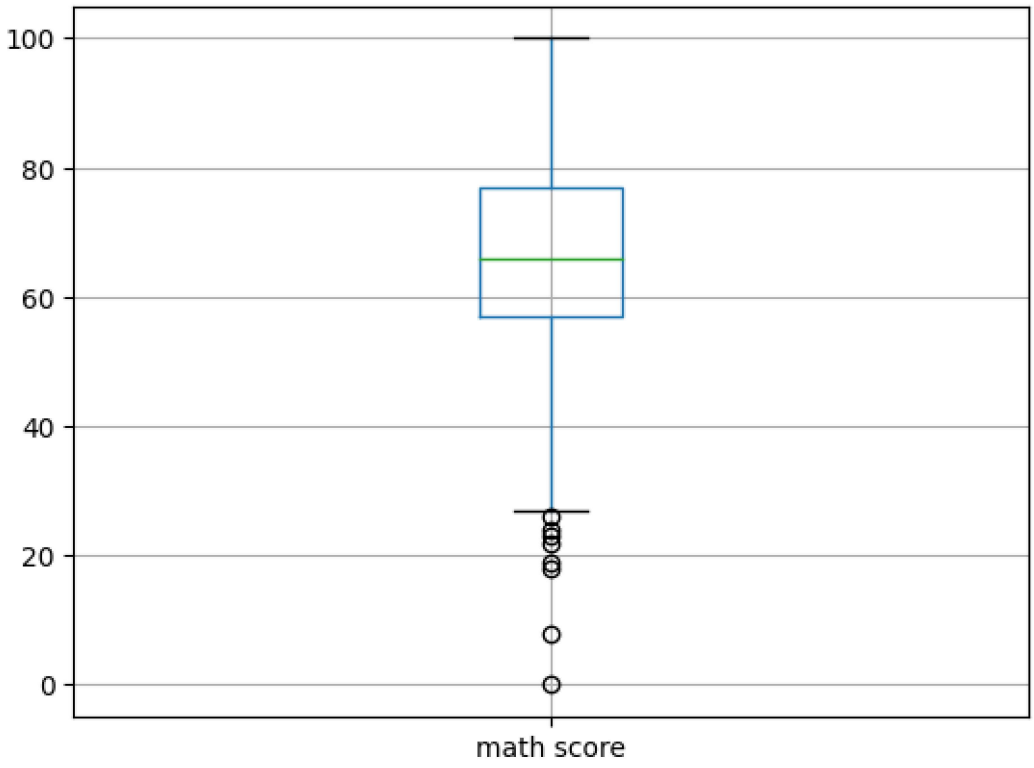
[ ]:
```
<Axes: xlabel='writing score', ylabel='Count'>
```

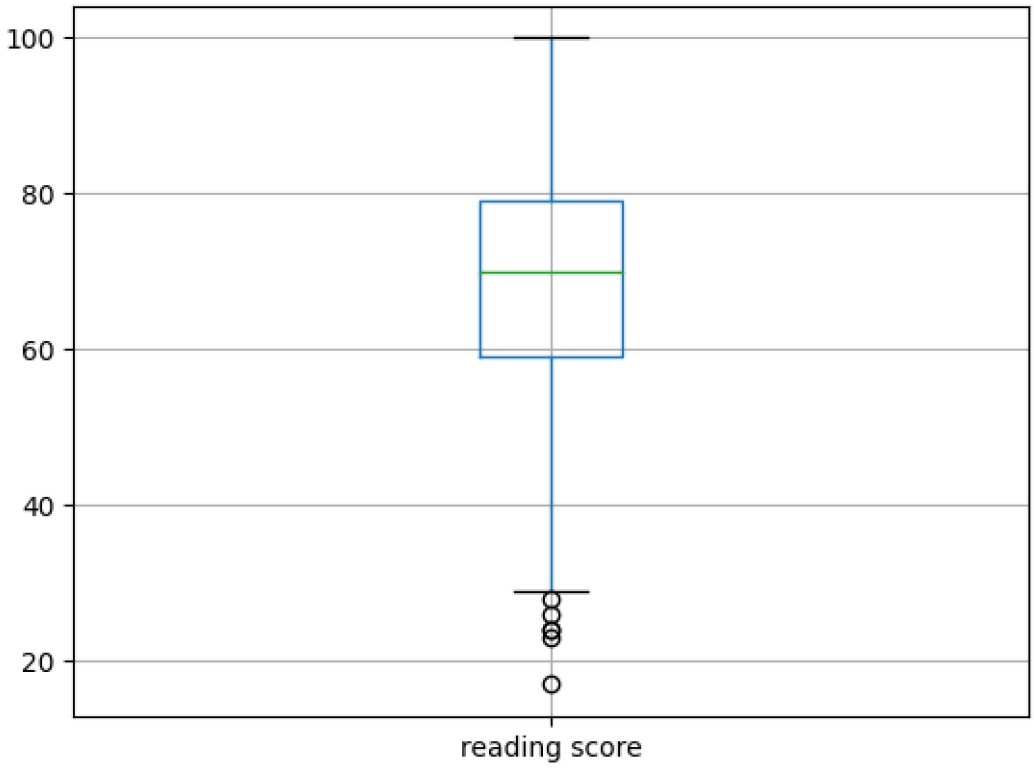


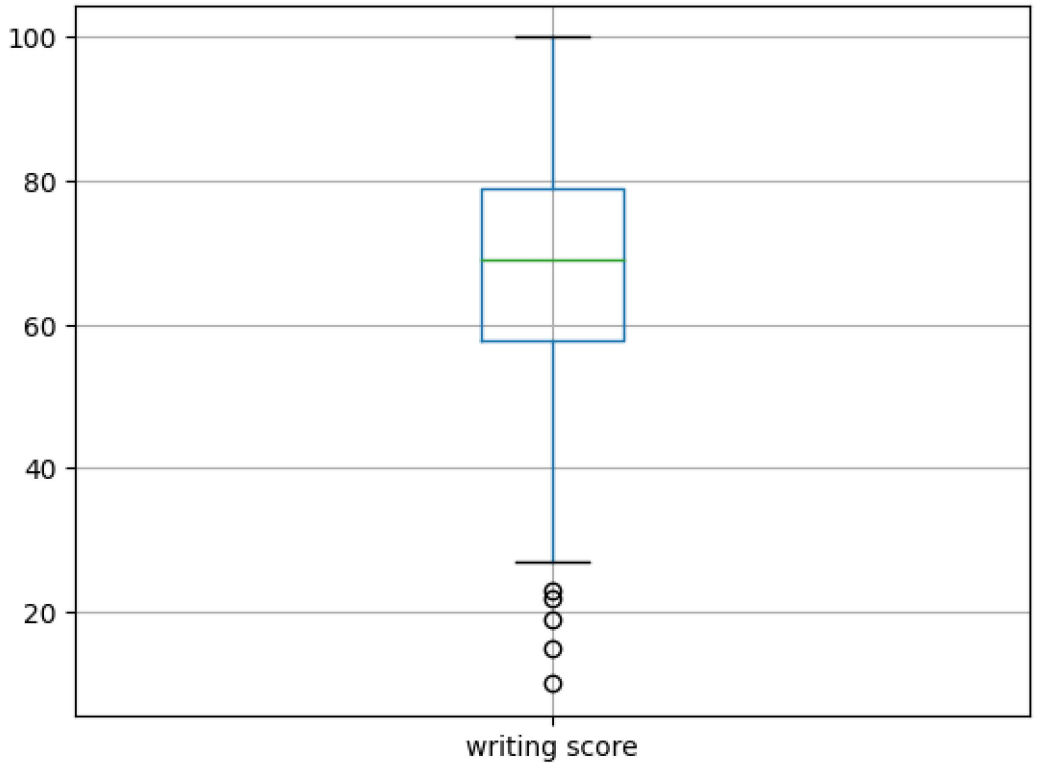
[ ]:
```
numeric_variable
```

[ ]:
```
Index(['math score', 'reading score', 'writing score'], dtype='object')
```

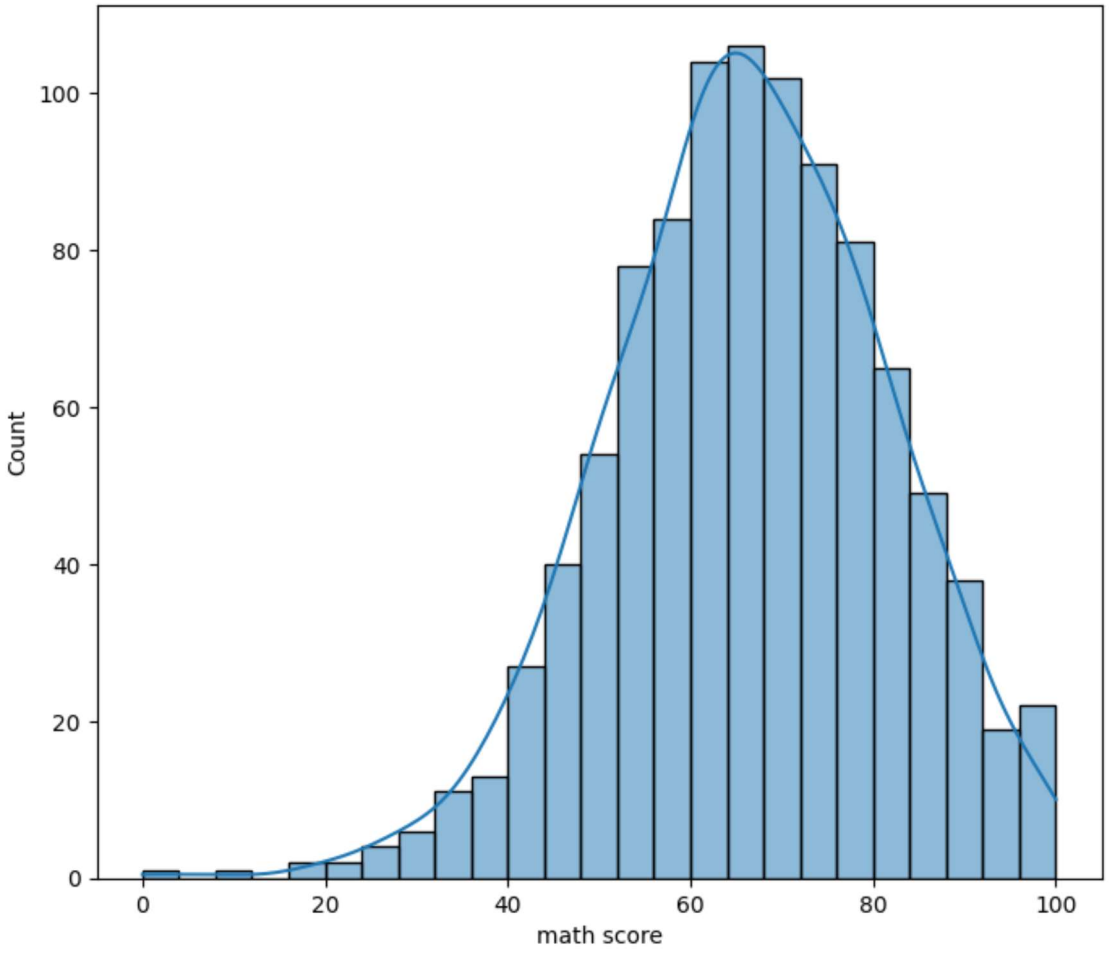
[ ]:
```
for i in numeric_variable:
  dataset.boxplot(i)
  plt.show()
```

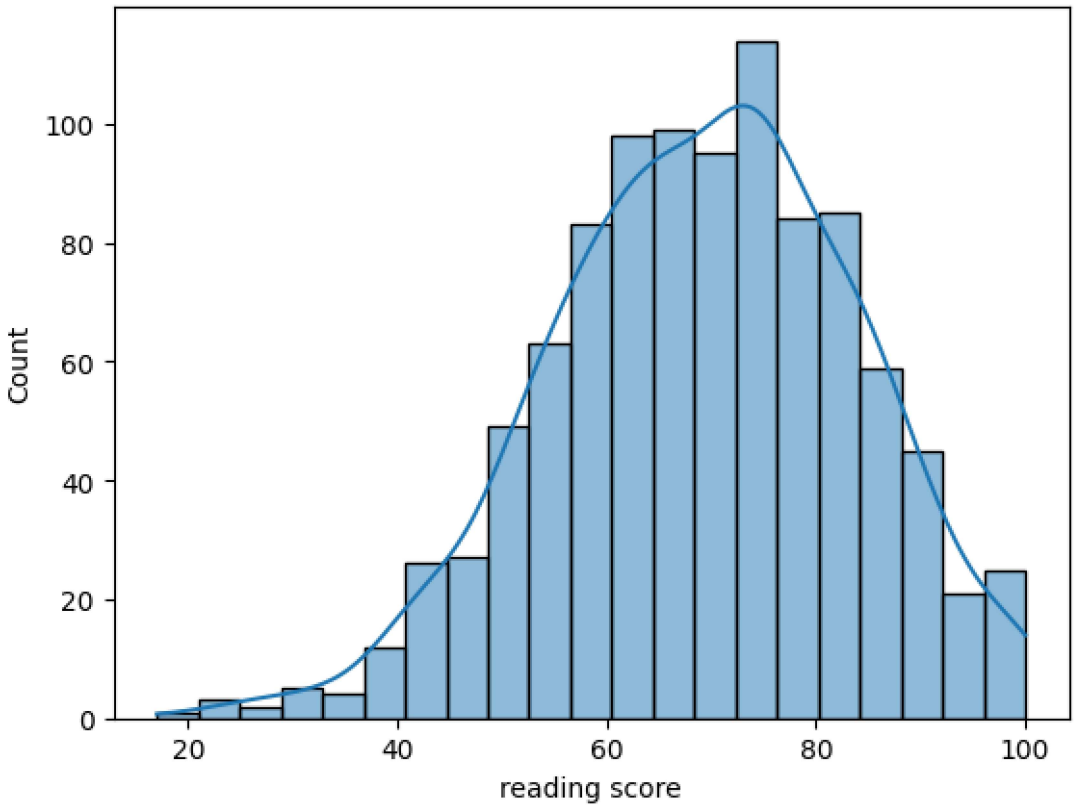

100
80
60
40
20
0
math score

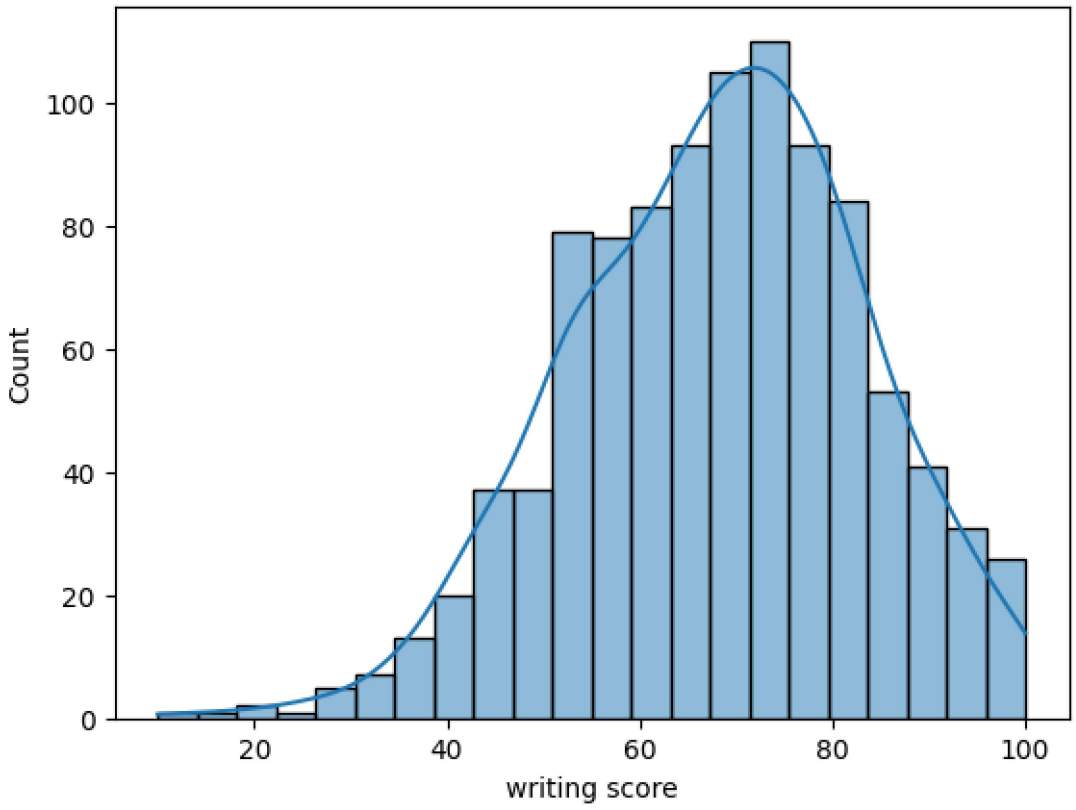

100
80
60
40
20
reading score



[ ]:
```
plt.figure(figsize=(8,7))
x = 0
for i in numeric_variable:
  sns.histplot(data=dataset,x=i,kde=True)
  plt.show()
```


Count
math score
0
20
40
60
80
100
0
20
40
60
80
100


Count
0
20
40
60
80
100
20
40
60
80
100
reading score




```
[ ]: upper_bound = dataset['math score'].median() + 3*dataset['math score'].std()
     lower_bound = dataset['math score'].median() - 3*dataset['math score'].std()
```

```
[ ]: upper_bound, lower_bound
```

```
[ ]: (111.48924028802836, 20.51075971197165)
```

```
[ ]: dataset['math score'].max()
```

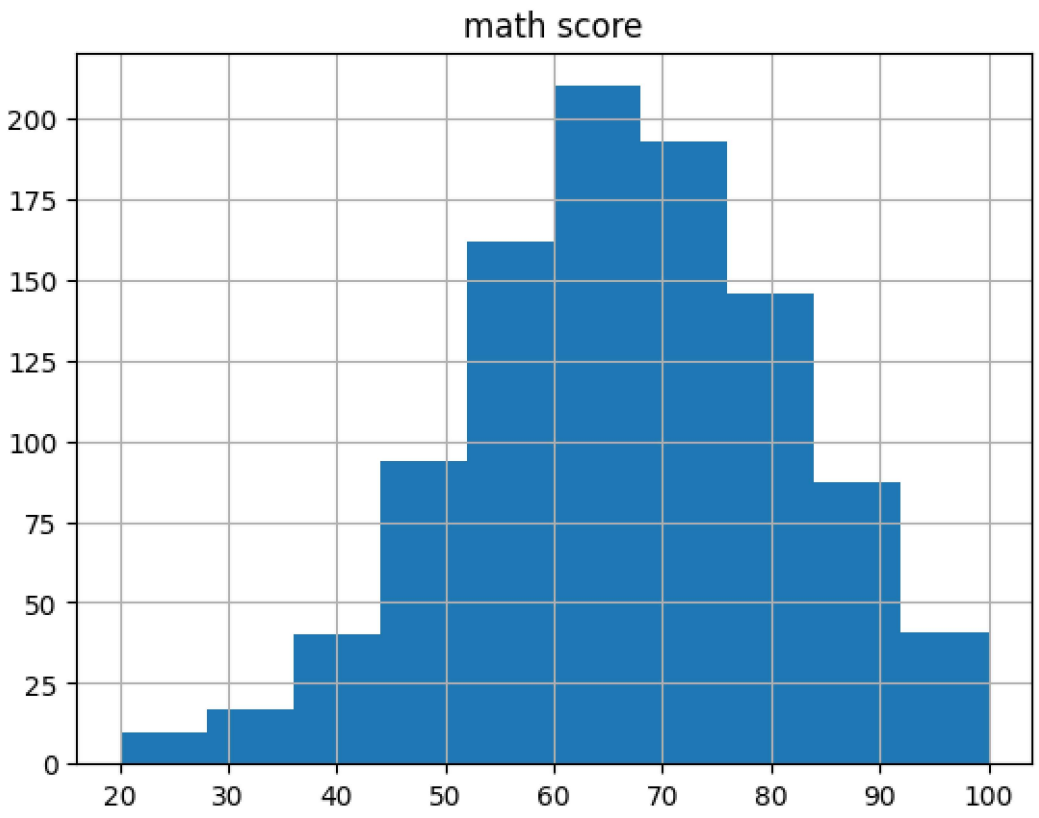
```
[ ]: 100
```

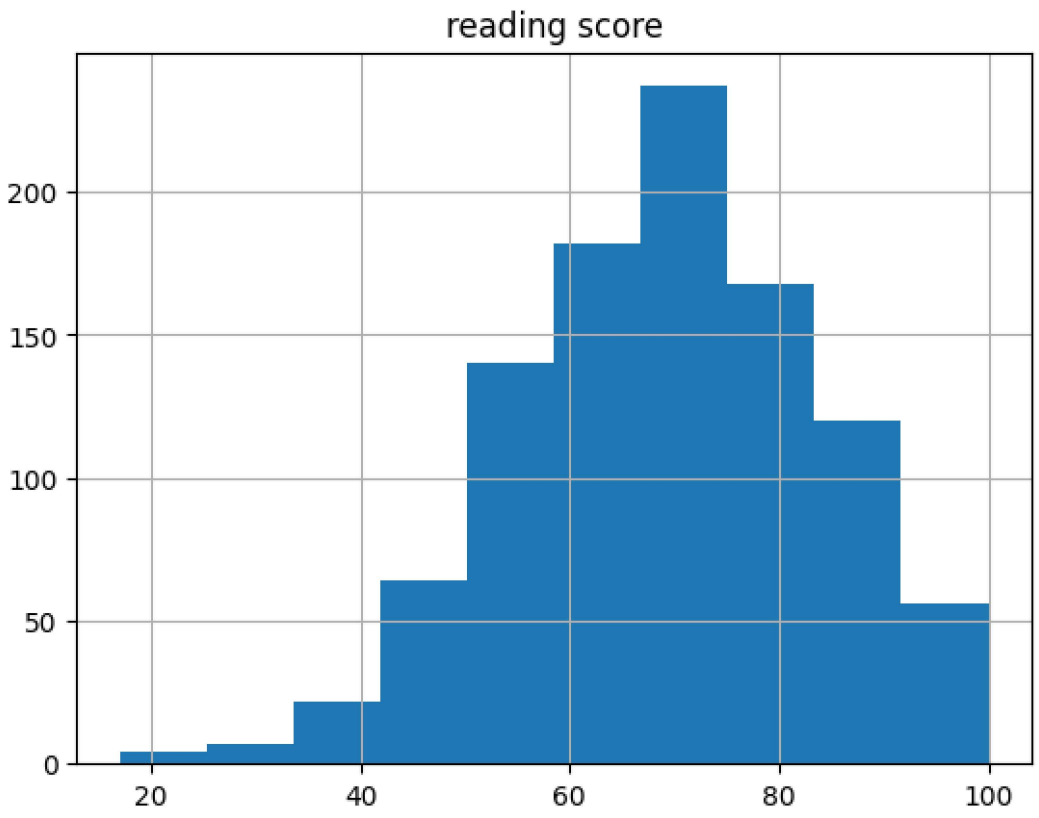
```
[ ]: dataset.loc[dataset['math score']<=20, 'math score'] = 20
```

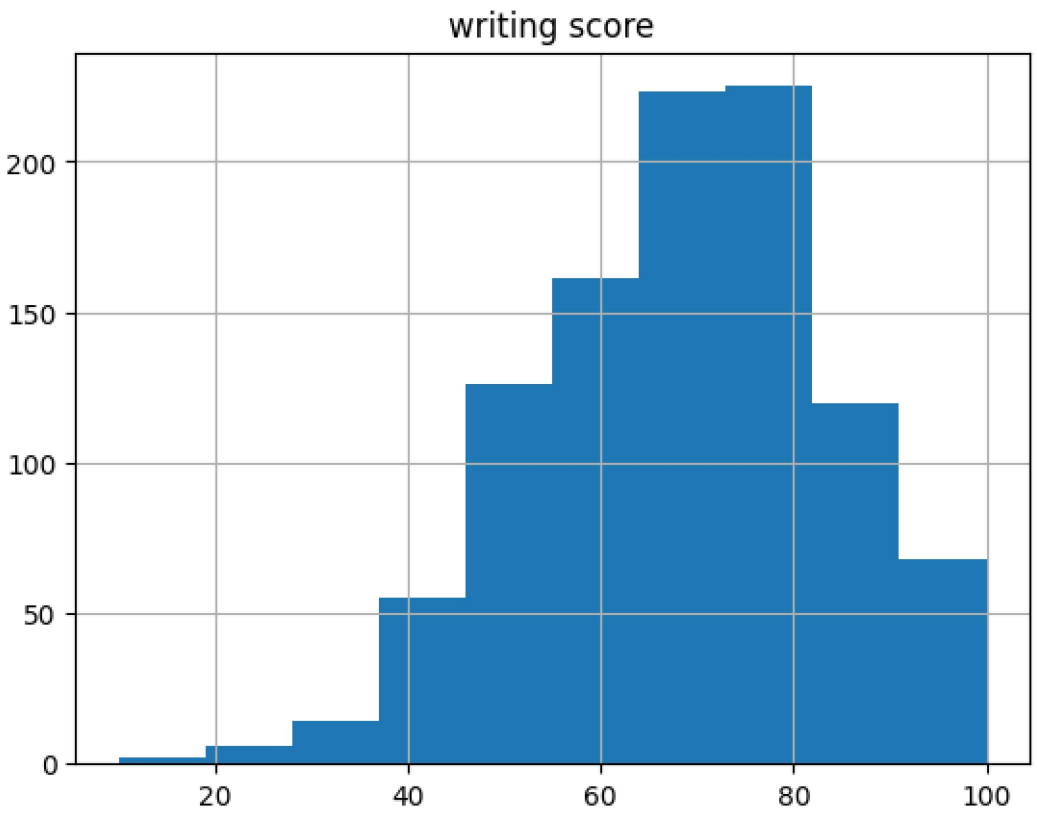
```
[ ]: numeric_variable
```

```
[ ]: Index(['math score', 'reading score', 'writing score'], dtype='object')
```

```
[ ]: for i in numeric_variable:
       dataset.hist(i)
```


math score
200
175
150
125
100
75
50
25
0
20
30
40
50
60
70
80
90
100


reading score
200
150
100
50
0
20
40
60
80
100


writing score
200
150
100
50
0
20
40
60
80
100

In [4]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [30]:
```
df = pd.read_csv("/home/pict/Documents/31437_DSBDA/Assign_03/Iris.csv")
```

In [31]:
```
df.head()
```

Out[31]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [32]:
```
df.tail()
```

Out[32]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [33]:
```
df.dtypes
```

Out[33]:
```
Id                int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [34]:
```
df['SepalLengthCm']
```

Out[34]:
```
0      5.1
1      4.9
2      4.7
3      4.6
```

```
4      5.0
      ...
145    6.7
146    6.3
147    6.5
148    6.2
149    5.9
Name: SepalLengthCm, Length: 150, dtype: float64
```

In [35]:
```
df['SepalLengthCm'].min()
```

Out[35]:
```
4.3
```

In [42]:
```
df['Species'].unique()
```

Out[42]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [46]:
```
group=df.groupby('Species')
```

In [51]:
```
iris_setosa=group.get_group('Iris-setosa')
iris_setosa
```

Out[51]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa
10	11	5.4	3.7	1.5	0.2	Iris-setosa
11	12	4.8	3.4	1.6	0.2	Iris-setosa

12	13	4.8	3.0	1.4	0.1	Iris-setosa
13	14	4.3	3.0	1.1	0.1	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa
15	16	5.7	4.4	1.5	0.4	Iris-setosa
16	17	5.4	3.9	1.3	0.4	Iris-setosa
17	18	5.1	3.5	1.4	0.3	Iris-setosa
18	19	5.7	3.8	1.7	0.3	Iris-setosa
19	20	5.1	3.8	1.5	0.3	Iris-setosa
20	21	5.4	3.4	1.7	0.2	Iris-setosa
21	22	5.1	3.7	1.5	0.4	Iris-setosa
22	23	4.6	3.6	1.0	0.2	Iris-setosa
23	24	5.1	3.3	1.7	0.5	Iris-setosa
24	25	4.8	3.4	1.9	0.2	Iris-setosa
25	26	5.0	3.0	1.6	0.2	Iris-setosa
26	27	5.0	3.4	1.6	0.4	Iris-setosa
27	28	5.2	3.5	1.5	0.2	Iris-setosa
28	29	5.2	3.4	1.4	0.2	Iris-setosa
29	30	4.7	3.2	1.6	0.2	Iris-setosa
30	31	4.8	3.1	1.6	0.2	Iris-setosa
31	32	5.4	3.4	1.5	0.4	Iris-setosa
32	33	5.2	4.1	1.5	0.1	Iris-setosa
33	34	5.5	4.2	1.4	0.2	Iris-setosa
34	35	4.9	3.1	1.5	0.1	Iris-

34	35	4.9	3.1	1.5	0.1	setosa
35	36	5.0	3.2	1.2	0.2	Iris-setosa
36	37	5.5	3.5	1.3	0.2	Iris-setosa
37	38	4.9	3.1	1.5	0.1	Iris-setosa
38	39	4.4	3.0	1.3	0.2	Iris-setosa
39	40	5.1	3.4	1.5	0.2	Iris-setosa
40	41	5.0	3.5	1.3	0.3	Iris-setosa
41	42	4.5	2.3	1.3	0.3	Iris-setosa
42	43	4.4	3.2	1.3	0.2	Iris-setosa
43	44	5.0	3.5	1.6	0.6	Iris-setosa
44	45	5.1	3.8	1.9	0.4	Iris-setosa
45	46	4.8	3.0	1.4	0.3	Iris-setosa
46	47	5.1	3.8	1.6	0.2	Iris-setosa
47	48	4.6	3.2	1.4	0.2	Iris-setosa
48	49	5.3	3.7	1.5	0.2	Iris-setosa
49	50	5.0	3.3	1.4	0.2	Iris-setosa

In [52]:
```
iris_setosa.describe()
```

Out[52]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	50.00000	50.00000	50.000000	50.000000	50.00000
mean	25.50000	5.00600	3.418000	1.464000	0.24400
std	14.57738	0.35249	0.381024	0.173511	0.10721
min	1.00000	4.30000	2.300000	1.000000	0.10000
25%	13.25000	4.80000	3.125000	1.400000	0.20000
50%	25.50000	5.00000	3.400000	1.500000	0.20000
75%	37.75000	5.20000	3.675000	1.575000	0.30000

In [15]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [16]:
```
boston = pd.read_csv("boston.csv")
```

In [17]:
```
print("------------Dataframe Info-----------------")
print(boston.info())
print("\n")
```

```
------------Dataframe Info-----------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  BLACK    506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
None
```

In [18]:
```
print("----------Dataframe Describe------------")
print(boston.describe())
print("\n")
```

```
----------Dataframe Describe------------
             CRIM          ZN       INDUS        CHAS         NOX
RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000
000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284
634
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702
617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561
000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885
500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208
500
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623
500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780
000

              AGE         DIS         RAD         TAX     PTRATIO        BL
ACK  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000
000
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674
032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294
864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320
000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377
500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440
000
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225
000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900
000

            LSTAT        MEDV
count  506.000000  506.000000
mean    12.653063   22.532806
std      7.141062    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

In [19]:
```
print("-----------Dataframe 5 Rows--------------")
print(boston.head())
print("\n")
```

```
-----------Dataframe 5 Rows--------------
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0

   PTRATIO   BLACK  LSTAT  MEDV
0     15.3  396.90   4.98  24.0
1     17.8  396.90   9.14  21.6
2     17.8  392.83   4.03  34.7
3     18.7  394.63   2.94  33.4
4     18.7  396.90   5.33  36.2
```

In [20]:
```
print("-----------Dataframe Columns List---------------")
print(boston.columns)
print("\n")
```

```
-----------Dataframe Columns List---------------
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'T
AX',
       'PTRATIO', 'BLACK', 'LSTAT', 'MEDV'],
      dtype='object')
```

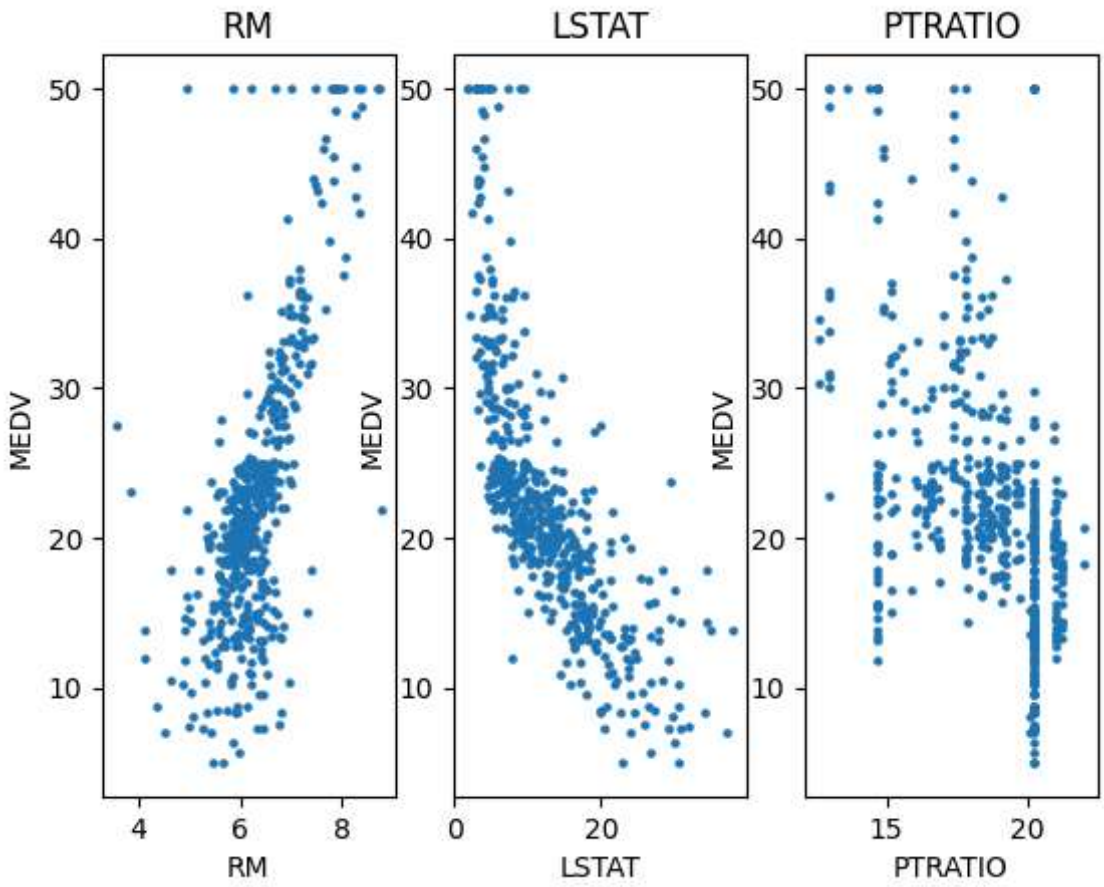
Selecting relevant features and target variable

In [21]:
```
X = boston[['RM', 'LSTAT', 'PTRATIO']]
y = boston['MEDV']
```

In [22]:
```
print("-------------Splitting data into training and test sets-------------
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
print("\n")
```

```
-------------Splitting data into training and test sets------------------
X_train shape: (404, 3)
X_test shape: (102, 3)
y_train shape: (404,)
y_test shape: (102,)
```

In [23]:
```
for i, feature in enumerate(X.columns):
    plt.subplot(1, 3, i + 1)
    plt.scatter(X[feature], y, marker='o', s=5)
    plt.title(feature)
    plt.xlabel(feature)
    plt.ylabel('MEDV')
```



In [24]:
```
plt.tight_layout()
plt.show()
```

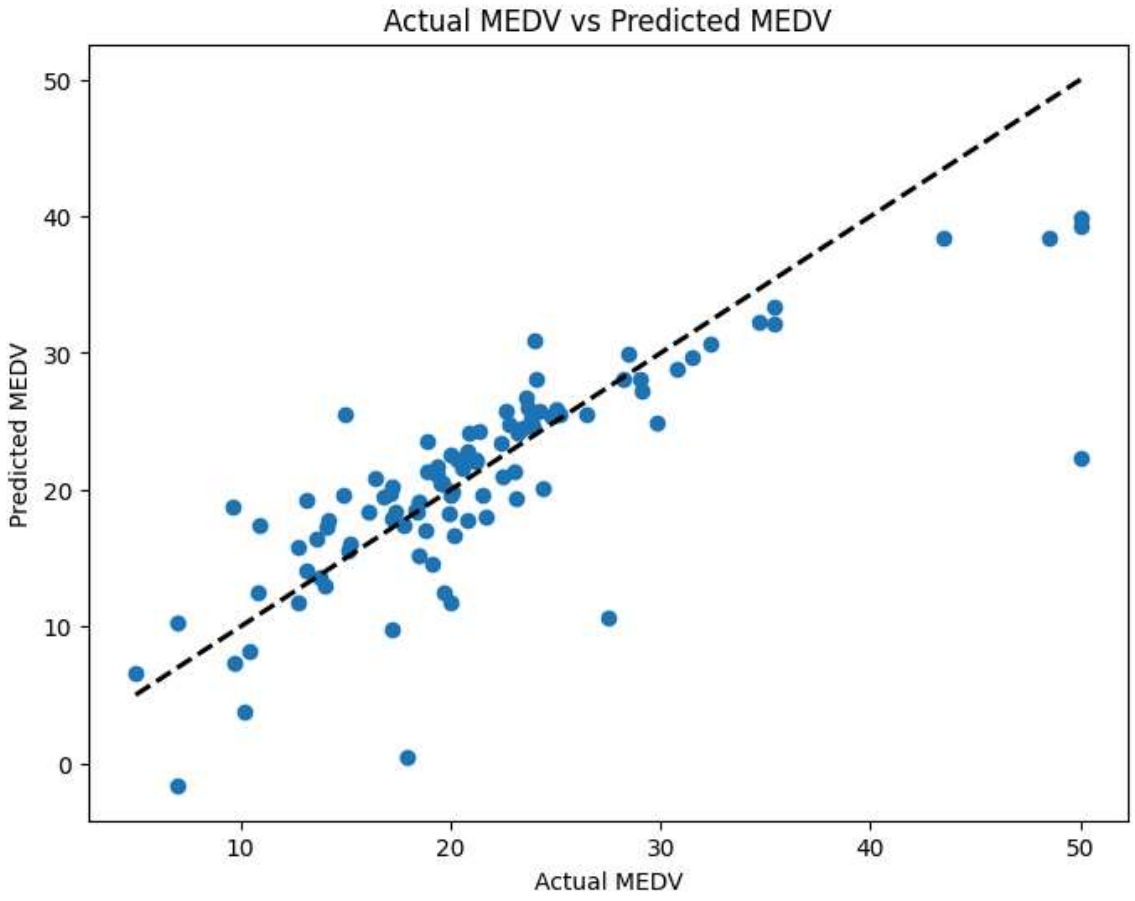
```
<Figure size 640x480 with 0 Axes>
```

Creating a linear regression model
Training the model
Making predictions

In [25]:
```
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

In [26]:
```
print("-------------Visualization after fitting model---------------")
# Visualization after fitting the model
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--',
plt.xlabel('Actual MEDV')
plt.ylabel('Predicted MEDV')
plt.title('Actual MEDV vs Predicted MEDV')
plt.show()
print("\n")
```

```
-------------Visualization after fitting model---------------
```



Evaluating the model

In [27]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("-----------Evaluation Result----------------")
print("Mean Squared Error:", mse)
print("R^2 Score:", r2)
print('\n')
```

```
-----------Evaluation Result----------------
Mean Squared Error: 27.114957415580573
R^2 Score: 0.6302528487272828
```

This code selects the 'RM' (average number of rooms per dwelling), 'LSTAT' (percentage of lower status of the population), and 'PTRATIO' (pupil-teacher ratio by town) columns as features and 'MEDV' (median value of owner-occupied homes) as the target variable. It then visualizes each feature against the target variable 'MEDV' before fitting the model and shows the predicted versus actual 'MEDV' values after fitting the model. Finally, it evaluates the model's performance using mean squared error and R-squared score.

In conclusion, this code loads the Boston housing dataset, explores it, splits it into training and test sets, trains a linear regression model, makes predictions on the test set, and evaluates the model performance using MSE and R^2 score. The visualizations before and after fitting the model provide insights into the data and model performance.

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Load Dataframe

In [3]:
```
df = pd.read_csv('Social_Network_Ads.csv')
```

In [4]:
```
print("----------------Dataframe Info-----------------")
print(df.info())
print("\n")
```

```
----------------Dataframe Info-----------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    float64
 3   EstimatedSalary  400 non-null    float64
 4   Purchased        400 non-null    int64
dtypes: float64(2), int64(2), object(1)
memory usage: 15.8+ KB
None
```

In [5]:
```
print("---------------Dataframe Descibe----------------")
print(df.describe())
print("\n")
```

```
---------------Dataframe Descibe----------------
            User ID         Age  EstimatedSalary   Purchased
count  4.000000e+02  400.000000       400.000000  400.000000
mean   1.569154e+07   37.655000     69742.500000    0.357500
std    7.165832e+04   10.482877     34096.960282    0.479864
min    1.556669e+07   18.000000     15000.000000    0.000000
25%    1.562676e+07   29.750000     43000.000000    0.000000
50%    1.569434e+07   37.000000     70000.000000    0.000000
75%    1.575036e+07   46.000000     88000.000000    1.000000
max    1.581524e+07   60.000000    150000.000000    1.000000
```

In [6]:
```
print("---------------First 5 rows of Dataframe----------------")
print(df.head())
print("\n")
```

```
---------------First 5 rows of Dataframe----------------
    User ID  Gender   Age  EstimatedSalary  Purchased
0  15624510    Male  19.0          19000.0          0
1  15810944    Male  35.0          20000.0          0
2  15668575  Female  26.0          43000.0          0
3  15603246  Female  27.0          57000.0          0
4  15804002    Male  19.0          76000.0          0
```

In [7]:
```
print("---------------Train Dataset-------------")
X = df[['Age', 'EstimatedSalary']]
Y = df['Purchased']
```

```
---------------Train Dataset-------------
```

In [8]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [9]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
print(f'Test  Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')
```

```
Train Dataset Size - X: (300, 2), Y: (300,)
Test  Dataset Size - X: (100, 2), Y: (100,)
```

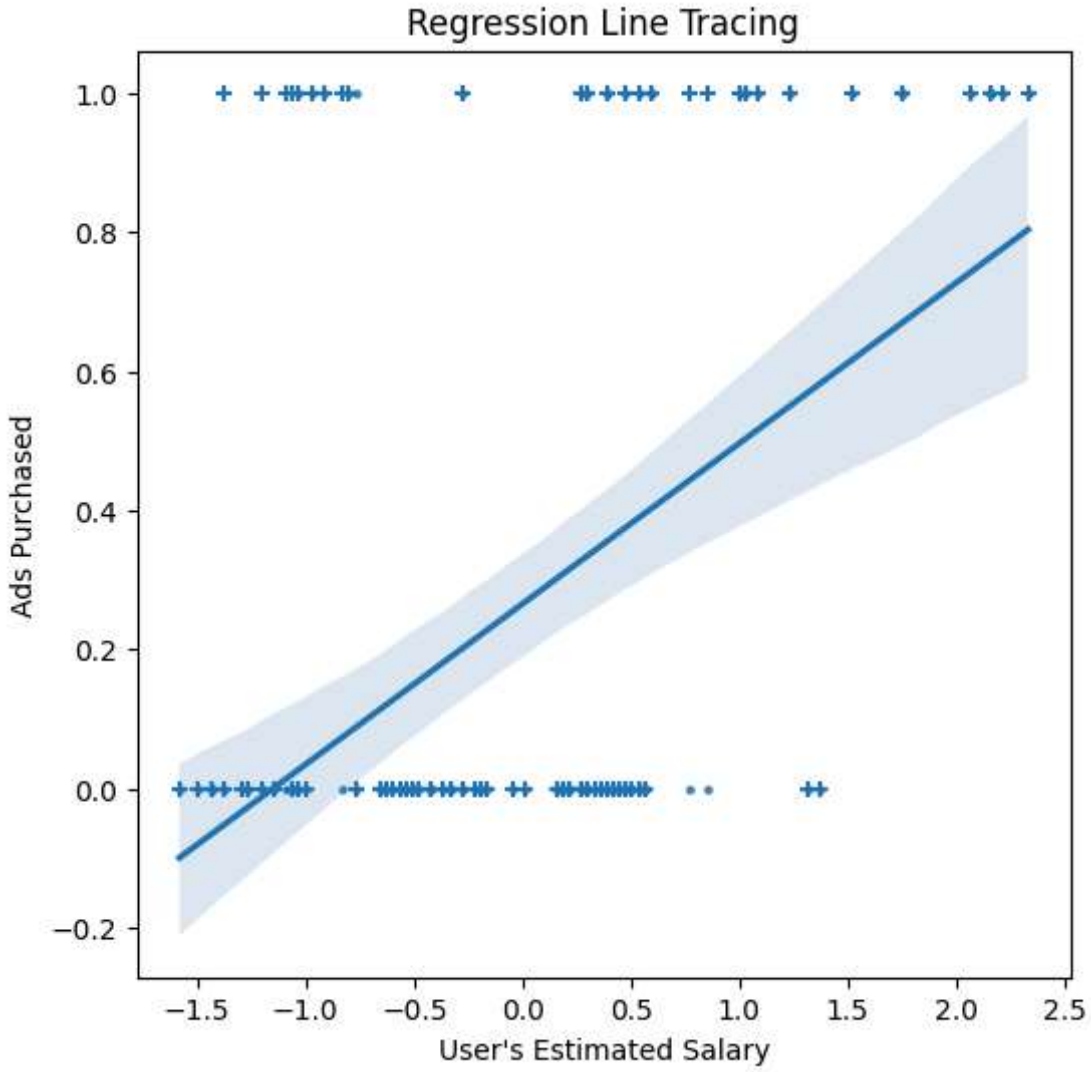
Linner Regression

This code fits a logistic regression model on the training data, makes predictions on the test data, and plots the regression line along with the actual test data to visualize the accuracy of the model.

In [10]:
```
print("----------------Linner Regression-------------------")
from sklearn.linear_model import LogisticRegression
```

```
----------------Linner Regression-------------------
```

In [11]:
```
lm = LogisticRegression(random_state=0, solver='lbfgs')
lm.fit(X_train, Y_train)
predictions = lm.predict(X_test)
plt.figure(figsize=(6, 6))
sns.regplot(x = X_test[:, 1], y = predictions, scatter_kws={'s':5})
plt.scatter(X_test[:, 1], Y_test, marker = '+')
plt.xlabel("User's Estimated Salary")
plt.ylabel('Ads Purchased')
plt.title('Regression Line Tracing')
plt.show()
```



In [12]:
```
print("-------------Confusion Matrix---------------")
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
-------------Confusion Matrix---------------
```

In [13]:
```
cm = confusion_matrix(Y_test, predictions)
print(f'''Confusion matrix :\n
               | Positive Prediction\t| Negative Prediction
---------------+-----------------------+---------------------
Positive Class | True Positive (TP) {cm[0, 0]}\t| False Negative (FN) {cm[0
---------------+-----------------------+---------------------
Negative Class | False Positive (FP) {cm[1, 0]}\t| True Negative (TN) {cm[1
```

```
Confusion matrix :

               | Positive Prediction    | Negative Prediction
---------------+------------------------+---------------------
Positive Class | True Positive (TP) 65  | False Negative (FN) 3
---------------+------------------------+---------------------
Negative Class | False Positive (FP) 8  | True Negative (TN) 24
```

In [14]:
```
cm = classification_report(Y_test, predictions)
print('Classification report : \n', cm)
```

```
Classification report :
               precision    recall  f1-score   support

           0       0.89      0.96      0.92        68
           1       0.89      0.75      0.81        32

    accuracy                           0.89       100
   macro avg       0.89      0.85      0.87       100
weighted avg       0.89      0.89      0.89       100
```

Visualizing Training set result

This code visualizes the training set results for the logistic regression model.
It creates a meshgrid of points, makes predictions on those points using the trained model, and plots the decision boundary.
It also scatters the actual training data points, coloring them based on their true class.
This allows us to see how well the logistic regression model fits the training data.

In [15]:
```
print("-----------------Visualizing Training set result----------------")
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, Y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[
```

```
-----------------Visualizing Training set result----------------
```

In [16]:
```
plt.figure(figsize=(9, 7.5))
plt.contourf(X1, X2, lm.predict(np.array([X1.ravel(), X2.ravel()]).T).resha
             alpha = 0.6, cmap = ListedColormap(('red', 'green')));
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                color = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
print("\n")
```

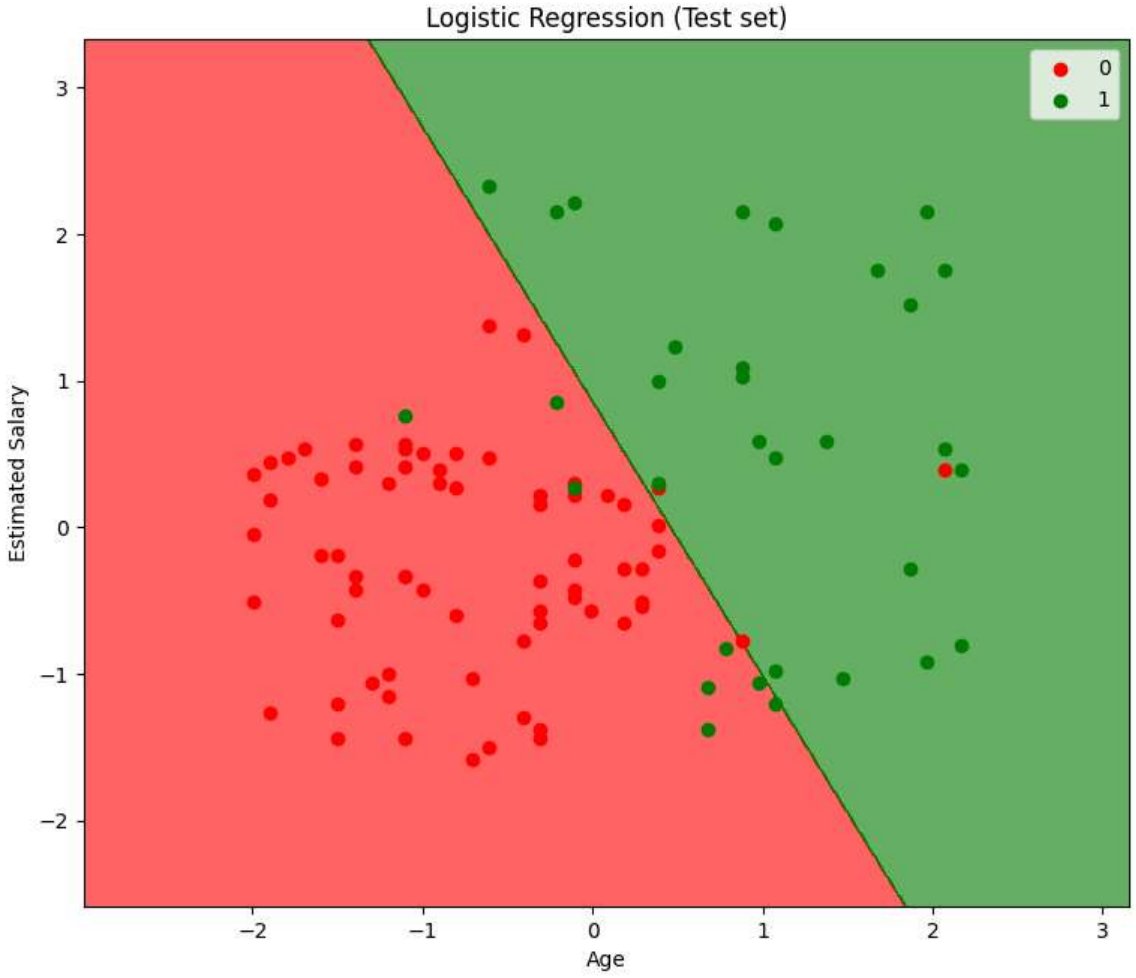
Visualizing Test set result

This code is visualizing the test set results after training the logistic regression model.
It creates meshgrid of points, makes predictions on those points using the trained model, and plots the decision boundary. It also scatters the actual test set data points on top.
This allows us to visualize how well the model is able to separate the two classes on the test data.
The various plot settings like colors, alpha, labels etc help interpret the results better.

In [17]:
```
print("-----------------Visualizing Test set result----------------")
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, Y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[
```

```
-----------------Visualizing Test set result----------------
```

In [18]:
```
plt.figure(figsize=(9, 7.5))
plt.contourf(X1, X2, lm.predict(np.array([X1.ravel(), X2.ravel()]).T).resha
             alpha = 0.6, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                color = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
print("\n")
```



In [18]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [19]:
```
df = pd.read_csv('iris.csv')
```

In [20]:
```
print("-----------Dataframe Info------------")
print(df.info())
print("\n")
```

```
-----------Dataframe Info------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

In [21]:
```
print("-----------Dataframe Describe------------")
print(df.describe())
print("\n")
```

```
-----------Dataframe Describe------------
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthC
m
count  150.000000     150.000000    150.000000     150.000000    150.00000
0
mean    75.500000       5.843333      3.054000       3.758667      1.19866
7
std     43.445368       0.828066      0.433594       1.764420      0.76316
1
min      1.000000       4.300000      2.000000       1.000000      0.10000
0
25%     38.250000       5.100000      2.800000       1.600000      0.30000
0
50%     75.500000       5.800000      3.000000       4.350000      1.30000
0
75%    112.750000       6.400000      3.300000       5.100000      1.80000
0
max    150.000000       7.900000      4.400000       6.900000      2.50000
0
```

In [22]:
```
print("----------Dataframe Head-----------")
print(df.head())
print("\n")
```

```
----------Dataframe Head-----------
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm Species
0   1            5.1           3.5            1.4           0.2  setosa
1   2            4.9           3.0            1.4           0.2  setosa
2   3            4.7           3.2            1.3           0.2  setosa
3   4            4.6           3.1            1.5           0.2  setosa
4   5            5.0           3.6            1.4           0.2  setosa
```

In [23]:
```
print("---------------Data Preprocessing----------------")
X = df.iloc[:,0:4]
Y = df['Species'].values
```

```
---------------Data Preprocessing---------------
```

In [24]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [25]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, ra
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

In [26]:
```
print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
print(f'Test  Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')
print("\n")
```

```
Train Dataset Size - X: (120, 4), Y: (120,)
Test  Dataset Size - X: (30, 4), Y: (30,)
```

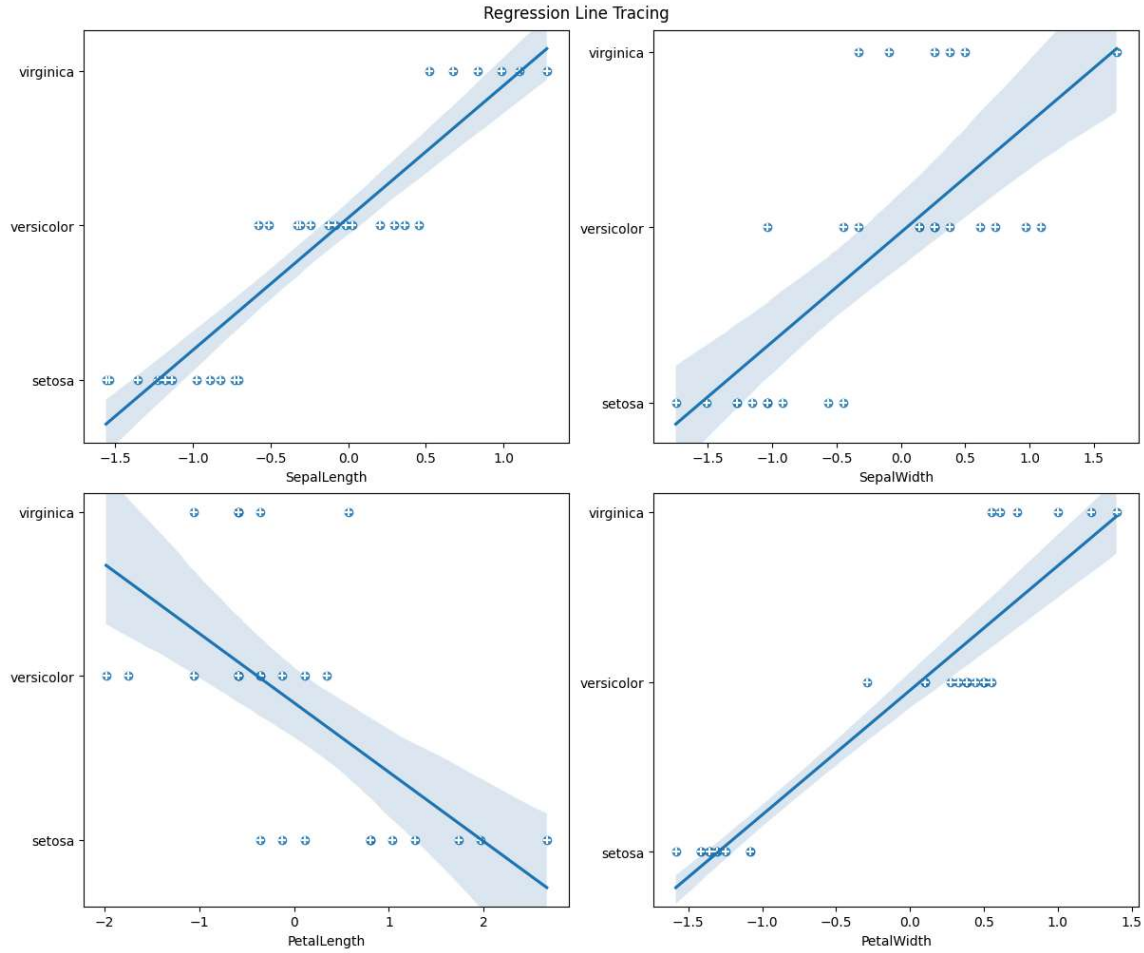
In [27]:
```
print("--------------Naive Bayes Classifier----------------------")
# This code fits a Naive Bayes classifier model on the training data, makes
# maps the predicted species labels to integers, and plots regression lines
# to the actual labels for each of the 4 feature columns. It shows how the
# predicting the species from each individual feature.
from sklearn.naive_bayes import GaussianNB
```

```
--------------Naive Bayes Classifier----------------------
```

In [28]:
```
classifier = GaussianNB()
classifier.fit(X_train, Y_train)
predictions = classifier.predict(X_test)
```

In [29]:
```
mapper = {'setosa': 0, 'versicolor': 1, 'virginica': 2}
predictions_ = [mapper[i] for i in predictions]
```

In [30]:
```
fig, axs = plt.subplots(2, 2, figsize = (12, 10), constrained_layout = True
_ = fig.suptitle('Regression Line Tracing')
for i in range(4):
    x, y = i // 2, i % 2
    _ = sns.regplot(x = X_test[:, i], y = predictions_, ax=axs[x, y])
    _ = axs[x, y].scatter(X_test[:, i][::-1], Y_test[::-1], marker = '+', c
    _ = axs[x, y].set_xlabel(df.columns[i + 1][:-2])
plt.show()
print("\n")
```



In [ ]:
```
```

In [31]:
```
print("------------Confusion Matrix-------------")
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
------------Confusion Matrix-------------
```

In [32]:
```
cm = confusion_matrix(Y_test, predictions)
print(f'''Confusion matrix :\n
               | Positive Prediction\t| Negative Prediction
---------------+-----------------------+---------------------
Positive Class | True Positive (TP) {cm[0, 0]}\t| False Negative (FN) {cm[0
---------------+-----------------------+---------------------
Negative Class | False Positive (FP) {cm[1, 0]}\t| True Negative (TN) {cm[1
```

```
Confusion matrix :

               | Positive Prediction    | Negative Prediction
---------------+------------------------+---------------------
Positive Class | True Positive (TP) 11  | False Negative (FN) 0
---------------+------------------------+---------------------
Negative Class | False Positive (FP) 0  | True Negative (TN) 13
```

In [33]:
```
cm = classification_report(Y_test, predictions)
print('Classification report : \n', cm)
```

```
Classification report :
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        11
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Sample Sentennces

In [10]:
```
sentence1 = "I will walk 500 miles and I would walk 500 more. Just to be the
            "a thousand miles to fall down at your door!"
sentence2 = "I played the play playfully as the players were playing in the
```

Tokenization

In [11]:
```
from nltk import word_tokenize, sent_tokenize
```

In [12]:
```
print("---------Tokenized Words------------")
print('Tokenized words:', word_tokenize(sentence1))
print("\n")
```

```
---------Tokenized Words------------
Tokenized words: ['I', 'will', 'walk', '500', 'miles', 'and', 'I', 'woul
d', 'walk', '500', 'more', '.', 'Just', 'to', 'be', 'the', 'man', 'who',
'walks', 'a', 'thousand', 'miles', 'to', 'fall', 'down', 'at', 'your', 'do
or', '!']
```

In [13]:
```
print("---------Tokenized Sentences------------")
print('Tokenized sentences:', sent_tokenize(sentence1))
print("\n")
```

```
---------Tokenized Sentences------------
Tokenized sentences: ['I will walk 500 miles and I would walk 500 more.',
'Just to be the man who walks a thousand miles to fall down at your doo
r!']
```

POS tagging

In [14]:
```
from nltk import pos_tag
print("---------POS Tagging------------")
token = word_tokenize(sentence1) + word_tokenize(sentence2)
print('POS tagged:', pos_tag(token))
print("\n")
```

```
---------POS Tagging------------
POS tagged: [('I', 'PRP'), ('will', 'MD'), ('walk', 'VB'), ('500', 'CD'),
('miles', 'NNS'), ('and', 'CC'), ('I', 'PRP'), ('would', 'MD'), ('walk',
'VB'), ('500', 'CD'), ('more', 'JJR'), ('.', '.'), ('Just', 'NNP'), ('to',
'TO'), ('be', 'VB'), ('the', 'DT'), ('man', 'NN'), ('who', 'WP'), ('walk
s', 'VBZ'), ('a', 'DT'), ('thousand', 'NN'), ('miles', 'NNS'), ('to', 'T
O'), ('fall', 'VB'), ('down', 'RP'), ('at', 'IN'), ('your', 'PRP$'), ('doo
r', 'NN'), ('!', '.'), ('I', 'PRP'), ('played', 'VBD'), ('the', 'DT'), ('p
lay', 'NN'), ('playfully', 'RB'), ('as', 'IN'), ('the', 'DT'), ('players',
'NNS'), ('were', 'VBD'), ('playing', 'VBG'), ('in', 'IN'), ('the', 'DT'),
('play', 'NN'), ('with', 'IN'), ('playfullness', 'NN')]
```

Stop-Words Removal

In [15]:
```
print("---------Stop-Words Removal------------")
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
token = word_tokenize(sentence1)
cleaned_token = []
for word in token:
    if word not in stop_words:
        cleaned_token.append(word)
print("Unclean version:", token)
print("\n")
print("Cleaned version:", cleaned_token)
print("\n")
```

```
---------Stop-Words Removal------------
Unclean version: ['I', 'will', 'walk', '500', 'miles', 'and', 'I', 'woul
d', 'walk', '500', 'more', '.', 'Just', 'to', 'be', 'the', 'man', 'who',
'walks', 'a', 'thousand', 'miles', 'to', 'fall', 'down', 'at', 'your', 'do
or', '!']


Cleaned version: ['I', 'walk', '500', 'miles', 'I', 'would', 'walk', '50
0', '.', 'Just', 'man', 'walks', 'thousand', 'miles', 'fall', 'door', '!']
```

In [16]:
```
print("---------Stemming------------")
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
token = word_tokenize(sentence2)
stemmed = [stemmer.stem(word) for word in token]
print("Stemmed words:", stemmed)
print("\n")
```

```
---------Stemming------------
Stemmed words: ['i', 'play', 'the', 'play', 'play', 'as', 'the', 'player',
'were', 'play', 'in', 'the', 'play', 'with', 'playful']
```

In [17]:
```
print("---------Lemmatization------------")
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
token = word_tokenize(sentence2)
lemmatized = [lemmatizer.lemmatize(word) for word in token]
print("Lemmatized words:", lemmatized)
print("\n")
```

```
---------Lemmatization------------
Lemmatized words: ['I', 'played', 'the', 'play', 'playfully', 'a', 'the',
'player', 'were', 'playing', 'in', 'the', 'play', 'with', 'playfullness']
```

In [1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

Load the Titanic dataset

In [2]:
```
titanic = sns.load_dataset('titanic')
```

Display the first few rows of the dataset

In [3]:
```
print("-------------Dataset first 5 rows-------------")
print(titanic.head())
print("\n")
```

```
-------------Dataset first 5 rows-------------
   survived  pclass     sex   age  sibsp  parch     fare embarked  class
\
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

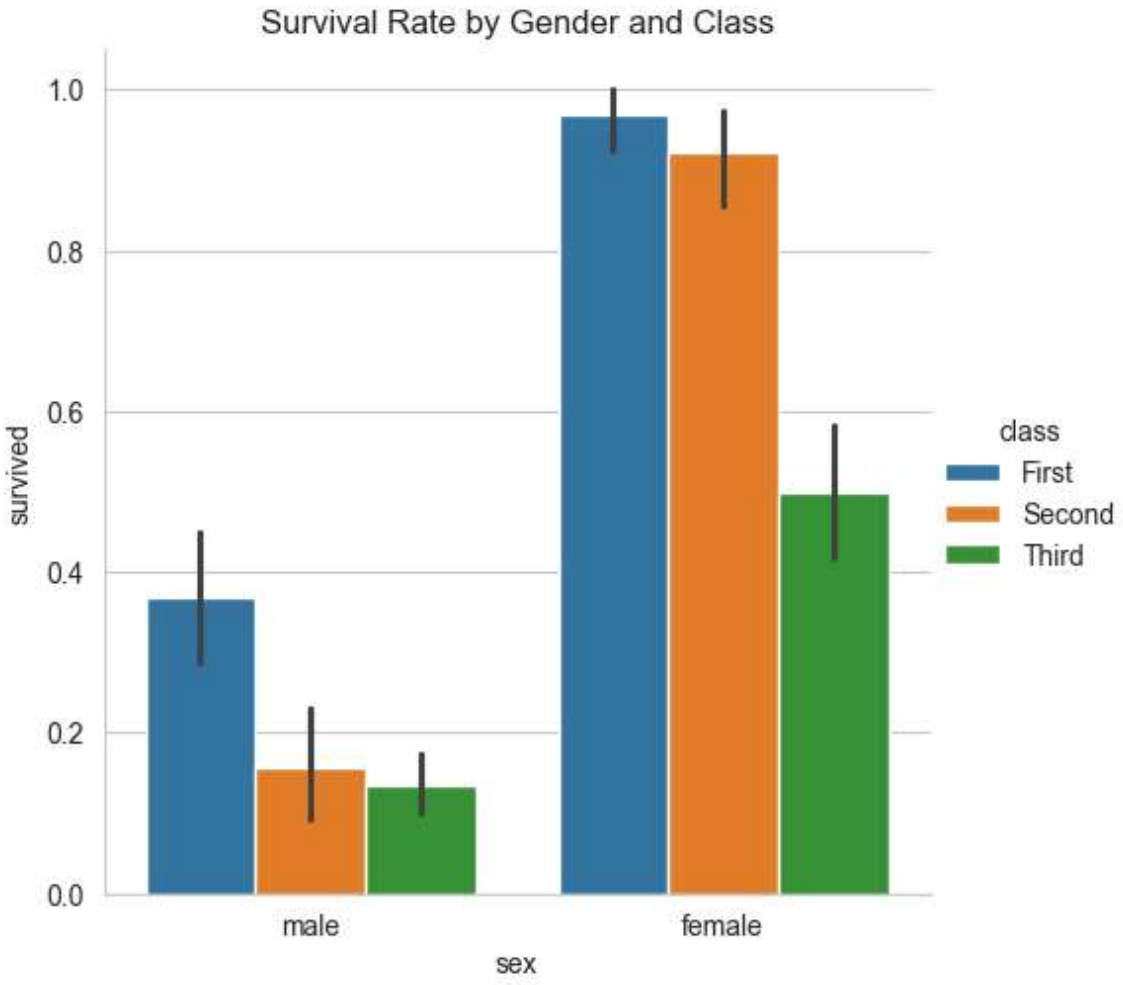
Use Seaborn to visualize patterns in the data

In [4]:
```
print("Setting style to whitegrid")
sns.set_style("whitegrid")
```

```
Setting style to whitegrid
```

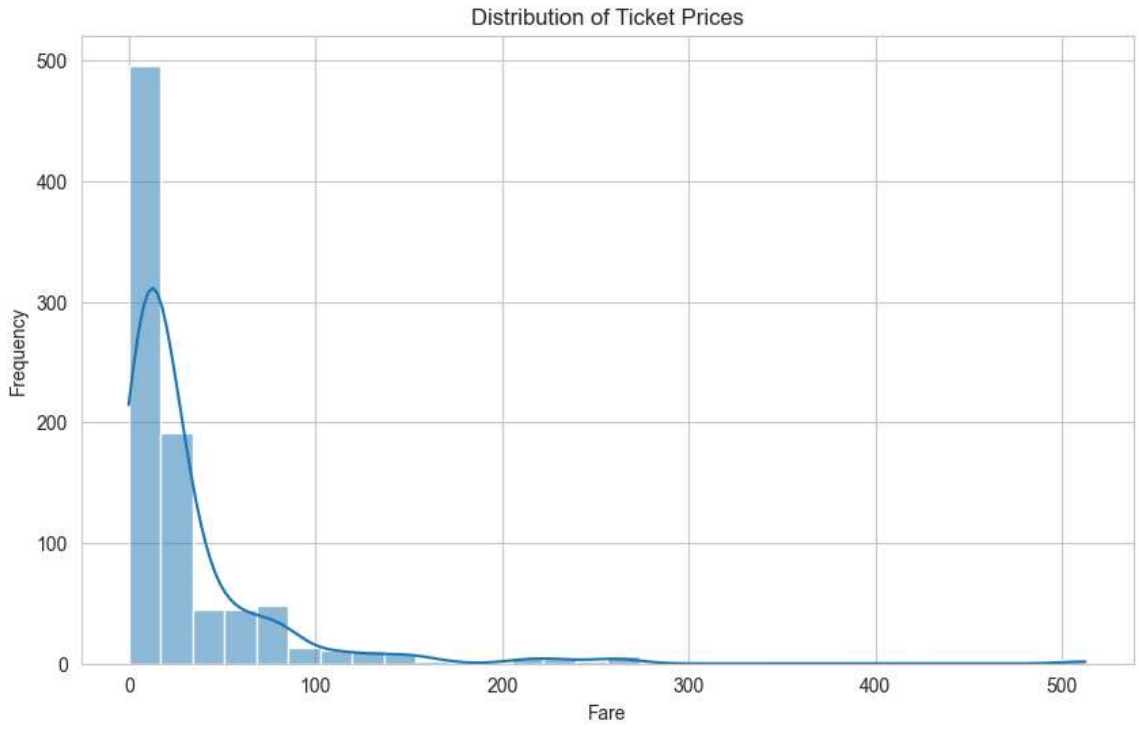
In [5]:
```
print("--------------Creating bar plot of survival rate by gender and class
sns.catplot(x="sex", y="survived", hue="class", kind="bar", data=titanic)
plt.title('Survival Rate by Gender and Class')
plt.show()
print("\n")
```

```
--------------Creating bar plot of survival rate by gender and class------
------------
```



In [6]:
```
print("--------------Plotting histogram of ticket prices-------------------
plt.figure(figsize=(10, 6))
sns.histplot(data=titanic, x='fare', bins=30, kde=True)
plt.title('Distribution of Ticket Prices')
plt.xlabel('Fare')
plt.ylabel('Frequency')
plt.show()
```

```
--------------Plotting histogram of ticket prices-------------------------
```



In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

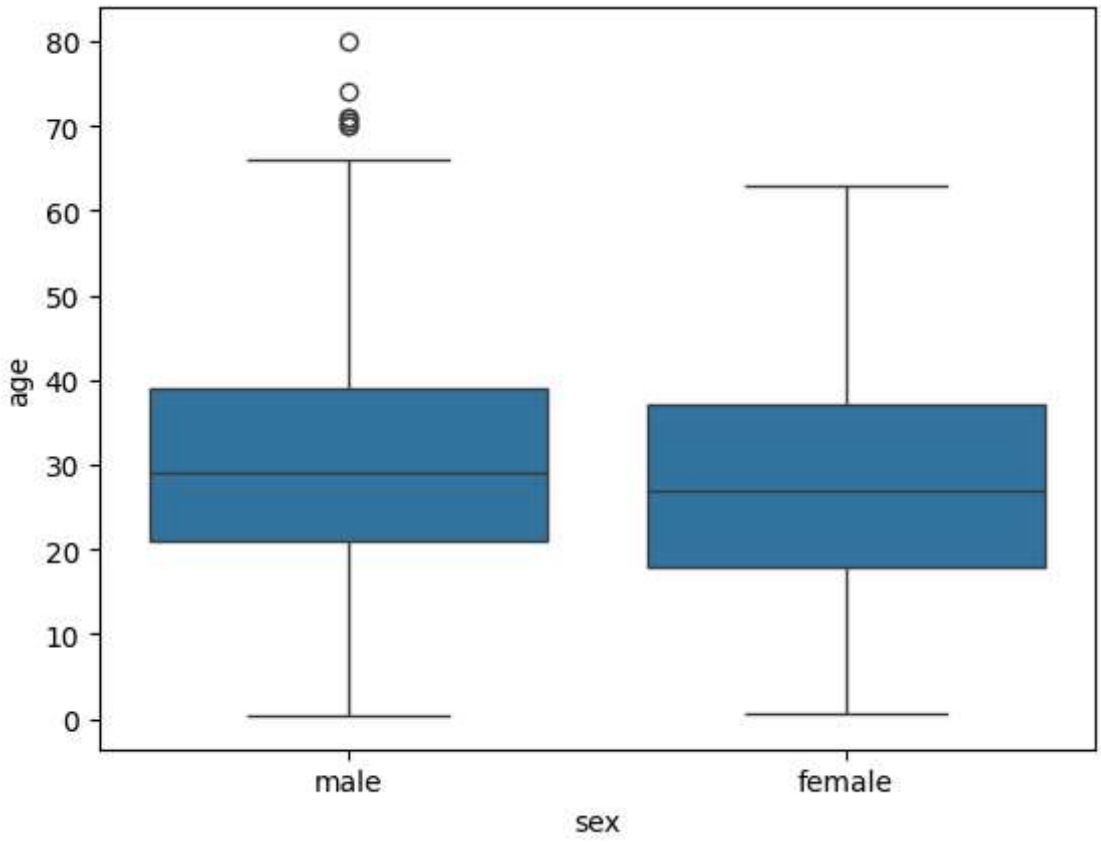
In [3]:
```
ds = sns.load_dataset('titanic')
```

In [4]:
```
print("-----------Dataset first 5 rows-------------")
print(ds.head())
print("\n")
```

```
-----------Dataset first 5 rows-------------
   survived  pclass     sex   age  sibsp  parch     fare embarked  class
\
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

In [5]:
```
print("-------Boxplot Gender vs Age---------------")
sns.boxplot(x='sex', y='age', data=ds)
plt.show()
print("\n")
```

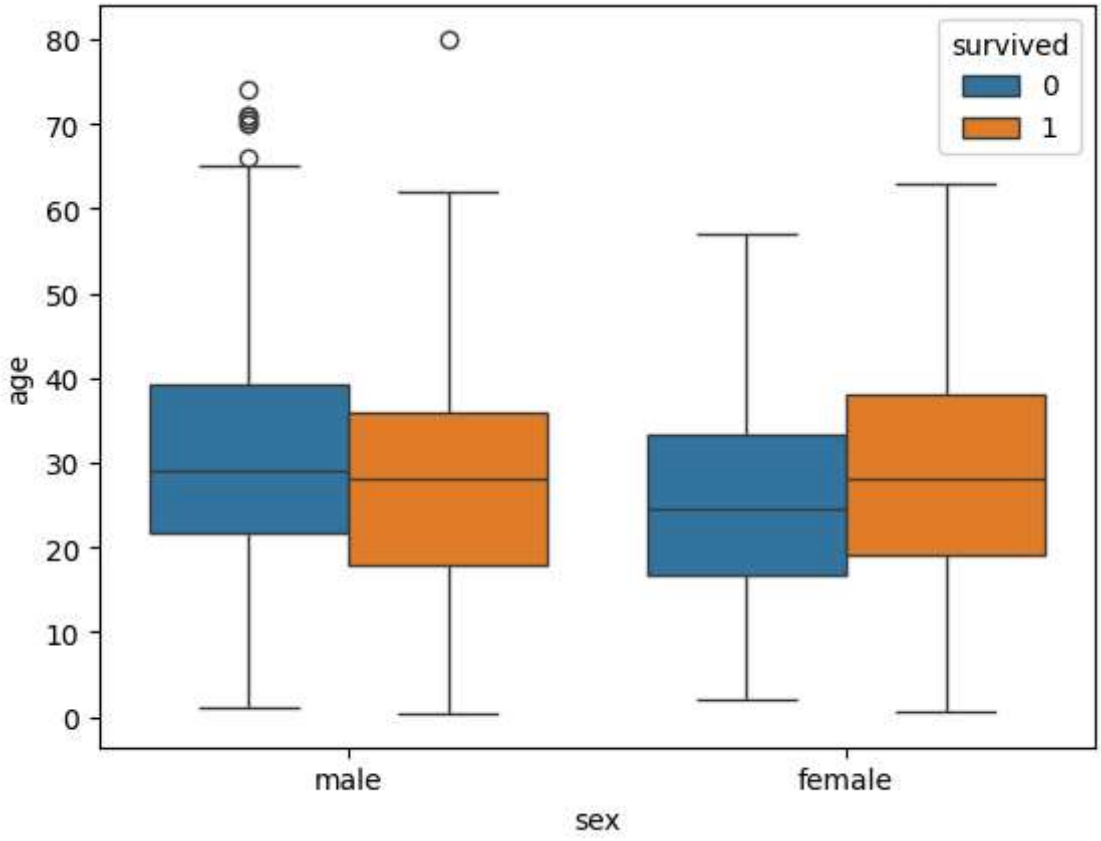
```
-------Boxplot Gender vs Age---------------
```



The first quartile starts at around 5 and ends at 22 which means that 25% of the passengers are aged between 5 and 25.
The second quartile starts at around 23 and ends at around 32 which means that 25% of the passengers are aged between 23 and 32.
Similarly, the third quartile starts and ends between 34 and 42, hence 25% passengers are aged within this range and finally the fourth or last quartile starts at 43 and ends around 65.

In [6]:
```
print("-------------Survived Passengers---------------------")
sns.boxplot(x='sex', y='age', data=ds, hue='survived')
plt.show()
print("\n")
```

```
-------------Survived Passengers---------------------
```



In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv("iris.data")
```

In [3]:
```
print("--------------Describe the Dataframe----------------------")
print(df.describe())
print("\n")
```

```
--------------Describe the Dataframe----------------------
              5.1         3.5         1.4         0.2
count  149.000000  149.000000  149.000000  149.000000
mean     5.848322    3.051007    3.774497    1.205369
std      0.828594    0.433499    1.759651    0.761292
min      4.300000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.600000    0.300000
50%      5.800000    3.000000    4.400000    1.300000
75%      6.400000    3.300000    5.100000    1.800000
max      7.900000    4.400000    6.900000    2.500000
```

In [4]:
```
print("--------------Shape of the Dataframe----------------------")
print(df.shape)
print("\n")
```

```
--------------Shape of the Dataframe----------------------
(149, 5)
```

In [5]:
```
print("--------------First 5 rows of the Dataframe----------------------")
print(df.head())
print("\n")
```

```
--------------First 5 rows of the Dataframe----------------------
   5.1  3.5  1.4  0.2  Iris-setosa
0  4.9  3.0  1.4  0.2  Iris-setosa
1  4.7  3.2  1.3  0.2  Iris-setosa
2  4.6  3.1  1.5  0.2  Iris-setosa
3  5.0  3.6  1.4  0.2  Iris-setosa
4  5.4  3.9  1.7  0.4  Iris-setosa
```

In [6]:
```
print("--------------Last 5 rows of the Dataframe----------------------")
print(df.tail())
print("\n")
```

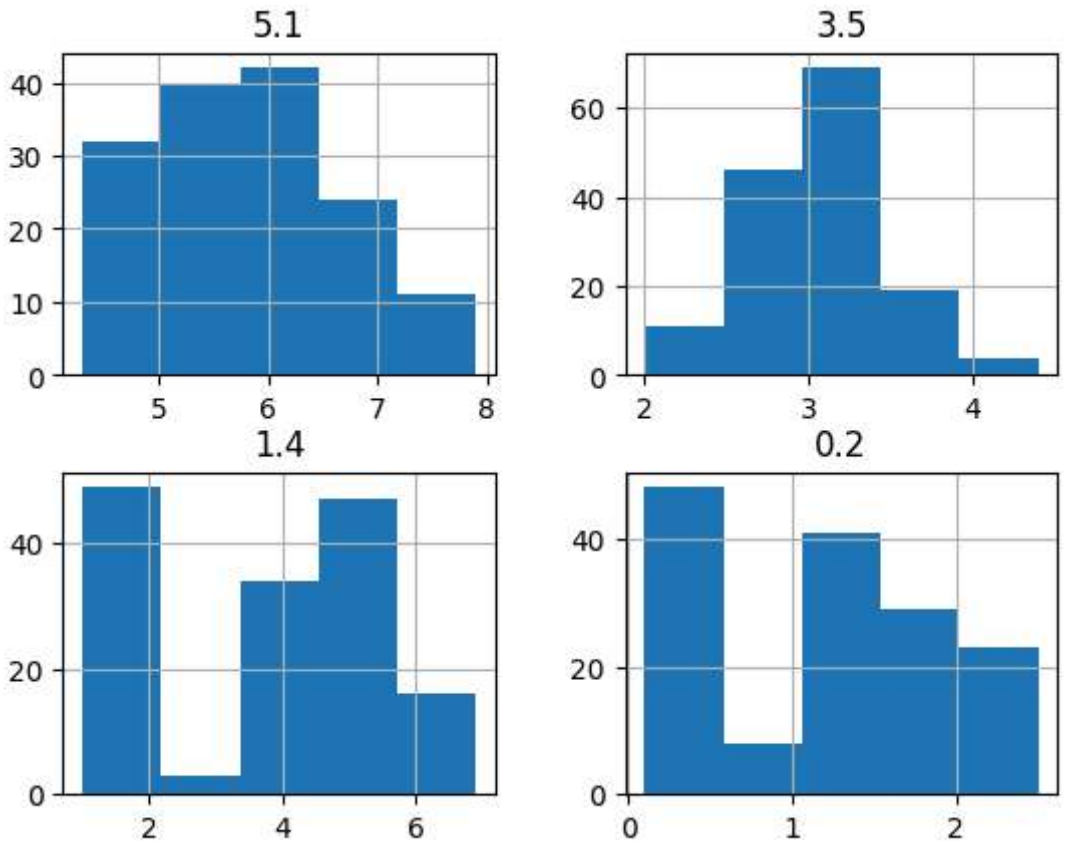
```
--------------Last 5 rows of the Dataframe----------------------
     5.1  3.5  1.4  0.2     Iris-setosa
144  6.7  3.0  5.2  2.3  Iris-virginica
145  6.3  2.5  5.0  1.9  Iris-virginica
146  6.5  3.0  5.2  2.0  Iris-virginica
147  6.2  3.4  5.4  2.3  Iris-virginica
148  5.9  3.0  5.1  1.8  Iris-virginica
```

In [7]:
```
print("--------------Mean of the First Column----------------------")
print(df["5.1"].mean())
print("\n")
```

```
--------------Mean of the First Column----------------------
5.8483221476510066
```

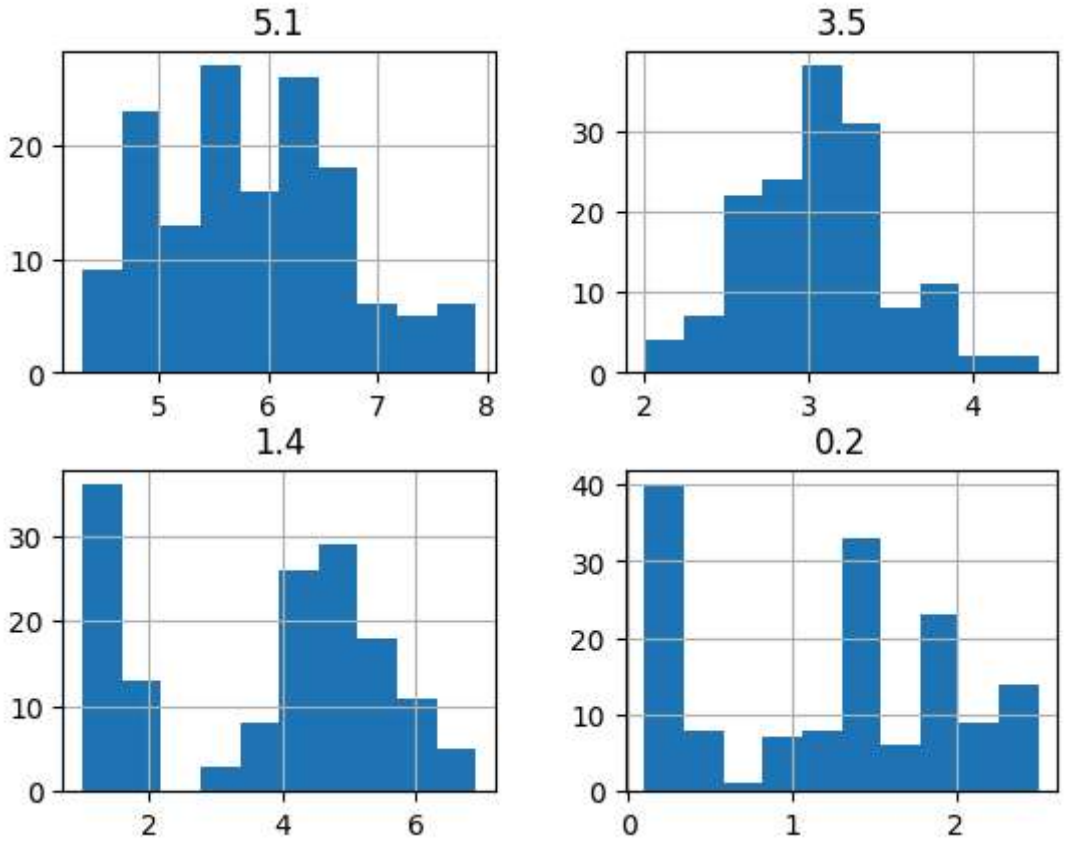
In [8]:
```
print("--------------Histogram of the Dataframe (using 5 bins)-------------
df.hist(bins=5)
plt.show()
print("\n")
```

```
--------------Histogram of the Dataframe (using 5 bins)------------------
---
```



In [9]:
```
print("-------------Histogram of the Dataframe---------------------")
df.hist()
plt.show()
print("\n")
```

```
-------------Histogram of the Dataframe---------------------
```



In [10]:
```
print("-------------Columns of the Dataframe---------------------")
print(df.columns)
print("\n")
```

```
-------------Columns of the Dataframe---------------------
Index(['5.1', '3.5', '1.4', '0.2', 'Iris-setosa'], dtype='object')
```

In [11]:
```
print("-------------Minimum value from Each Column---------------------")
print(df.min())
print("\n")
```

```
-------------Minimum value from Each Column---------------------
5.1                  4.3
3.5                  2.0
1.4                  1.0
0.2                  0.1
Iris-setosa    Iris-setosa
dtype: object
```

In [12]:
```
print("--------------Maximum value from Each Column----------------------")
print(df.max())
print("\n")
```

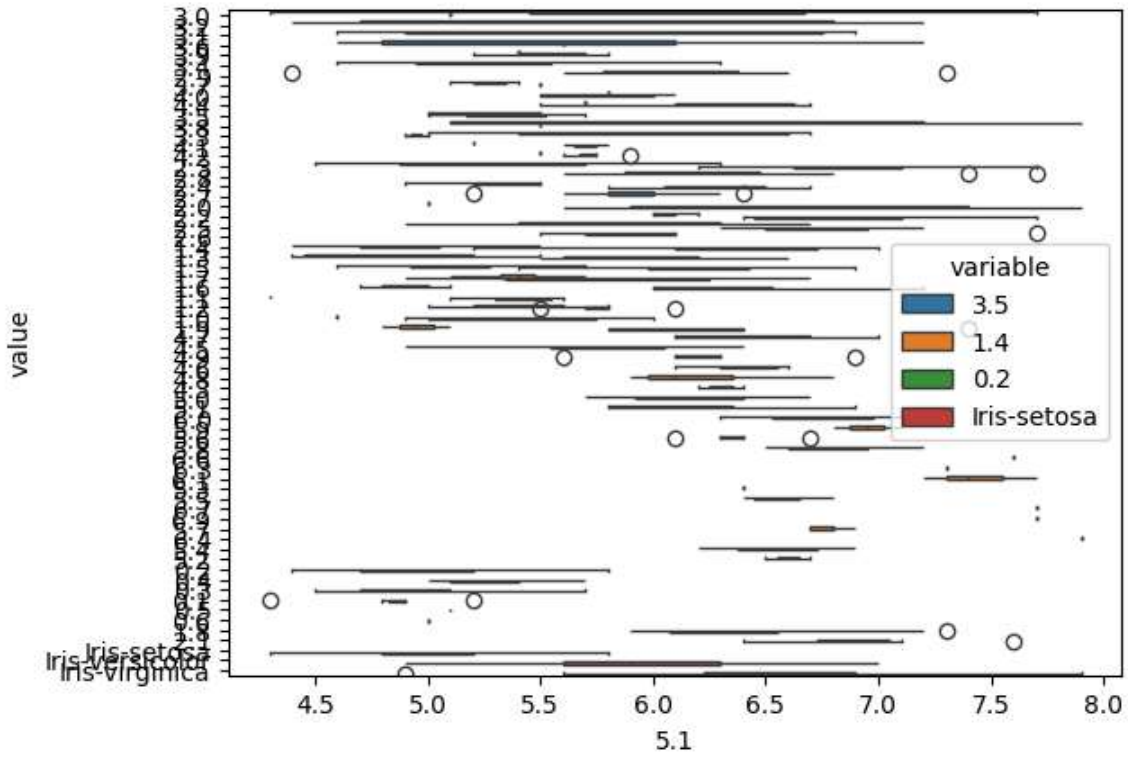
```
--------------Maximum value from Each Column----------------------
5.1                         7.9
3.5                         4.4
1.4                         6.9
0.2                         2.5
Iris-setosa    Iris-virginica
dtype: object
```

In [13]:
```
print("--------------Quantile of the Dataframe----------------------")
print(df.quantile([0, 0.25, 0.5, 0.75, 1.0], numeric_only=True))
print("\n")
```

```
--------------Quantile of the Dataframe----------------------
      5.1  3.5  1.4  0.2
0.00  4.3  2.0  1.0  0.1
0.25  5.1  2.8  1.6  0.3
0.50  5.8  3.0  4.4  1.3
0.75  6.4  3.3  5.1  1.8
1.00  7.9  4.4  6.9  2.5
```

In [14]:
```
print("--------------Correlation of the Dataframe----------------------")
iris_long = pd.melt(df, id_vars='5.1')
ax = sns.boxplot(x="5.1", y="value", hue="variable", data=iris_long)
plt.show()
```

```
--------------Correlation of the Dataframe----------------------
```

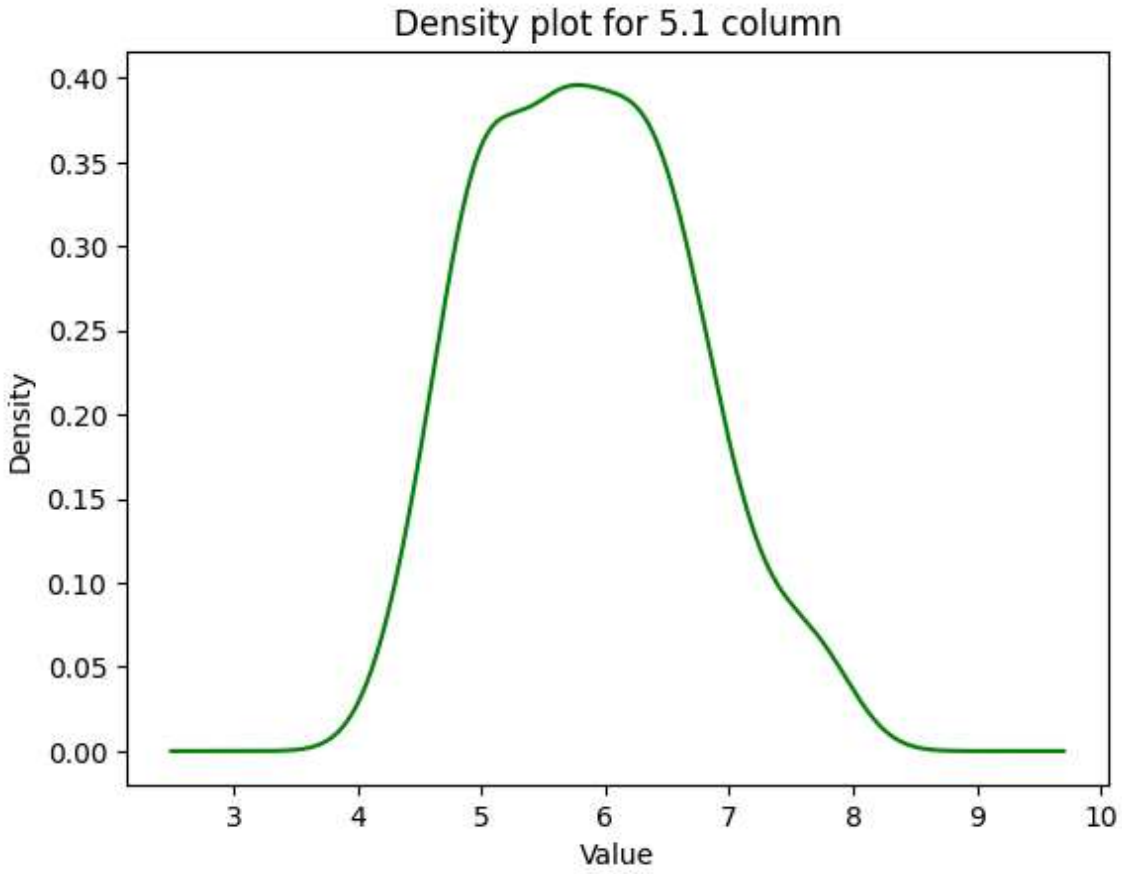


In [15]:
```
print("--------------Frequecy of each value in the first column------------")
print(df['5.1'].value_counts())
print("\n")
```

```
--------------Frequecy of each value in the first column------------------
----
5.1
5.0    10
6.3     9
6.7     8
5.1     8
5.7     8
5.8     7
5.5     7
6.4     7
5.6     6
5.4     6
6.1     6
6.0     6
4.9     6
6.5     5
4.8     5
6.2     4
7.7     4
6.9     4
5.2     4
4.6     4
4.4     3
5.9     3
7.2     3
6.8     3
4.7     2
6.6     2
4.3     1
7.0     1
4.5     1
7.1     1
7.6     1
7.3     1
5.3     1
7.4     1
7.9     1
Name: count, dtype: int64
```

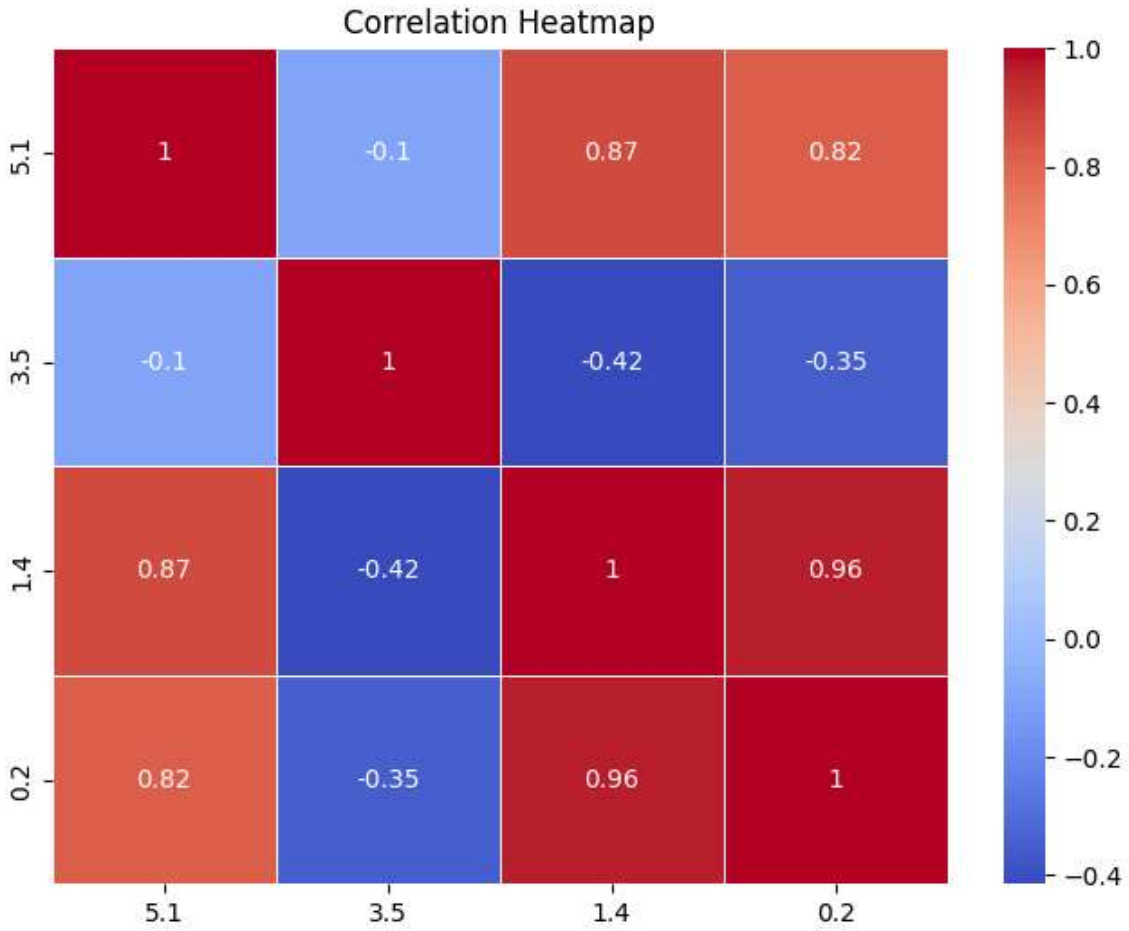
In [16]:
```
print("--------------Density plot for 5.1 column----------------------")
df['5.1'].plot.density(color='green')
plt.title('Density plot for 5.1 column')
plt.xlabel('Value')
plt.ylabel('Density')
plt.show()
print("\n")
```

```
--------------Density plot for 5.1 column----------------------
```



In [17]:
```
print("--------------Heatmap for the Correlation----------------------")
subset_df = df.iloc[:, :4]
plt.figure(figsize=(8, 6))
sns.heatmap(subset_df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```

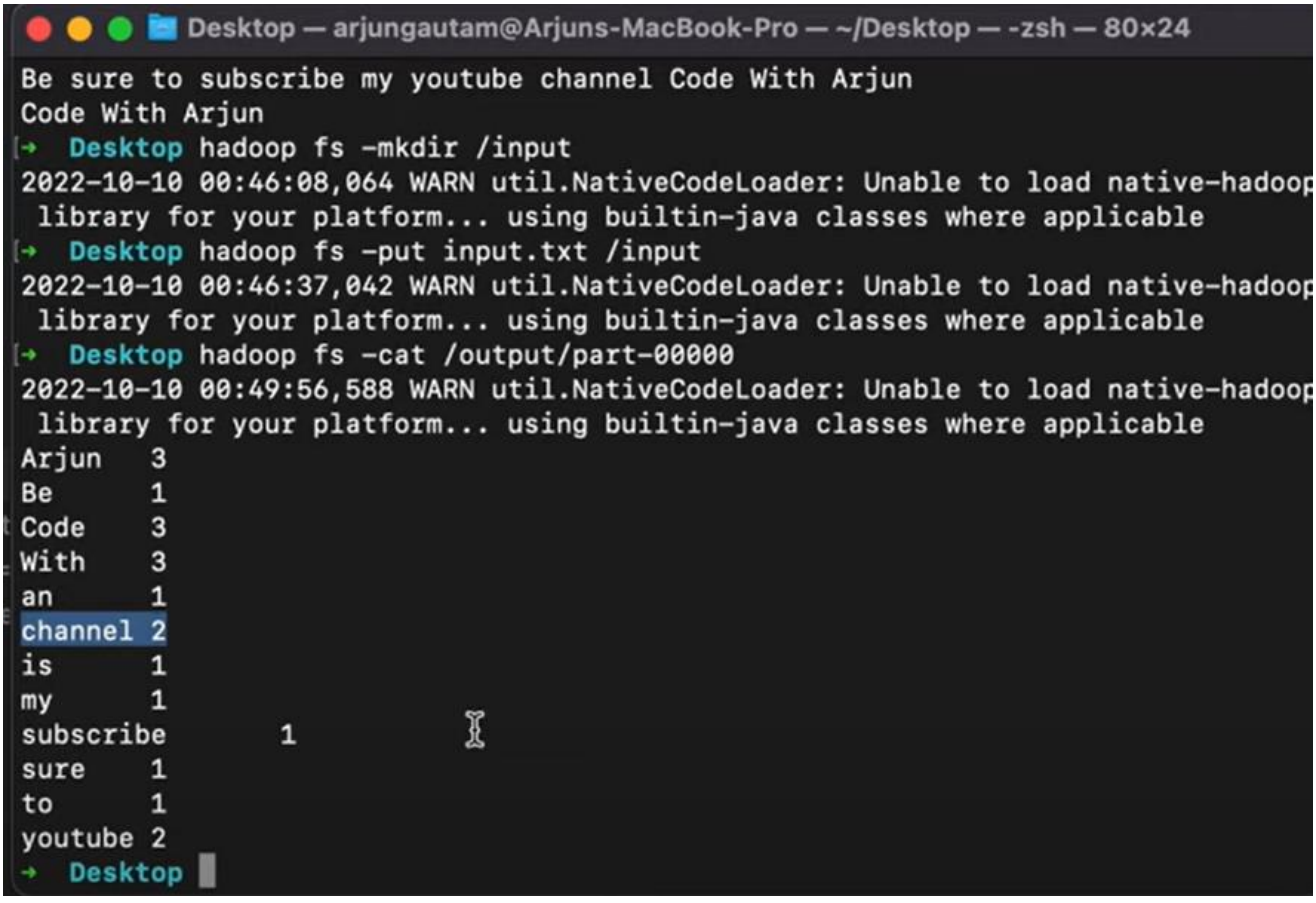
```
--------------Heatmap for the Correlation----------------------
```



```
//dsabda 11
//WC_Mapper.java
package org.codewitharjun;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WC_Mapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,IntWritable>{
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
   public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,
                Reporter reporter) throws IOException{
      String line = value.toString();
      StringTokenizer  tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()){
         word.set(tokenizer.nextToken());
         output.collect(word, one);
      }
   }

}

//WC_Reducer.java

package org.codewitharjun;

import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WC_Reducer  extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable> {
   public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,
                   Reporter reporter) throws IOException {
      int sum=0;
      while (values.hasNext()) {
         sum+=values.next().get();
      }
      output.collect(key,new IntWritable(sum));
   }
}
```

```
//WC_Runner.java
package org.codewitharjun;
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
public class WC_Runner {
   public static void main(String[] args) throws IOException{
      JobConf conf = new JobConf(WC_Runner.class);
      conf.setJobName("WordCount");
      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(IntWritable.class);
      conf.setMapperClass(WC_Mapper.class);
      conf.setCombinerClass(WC_Reducer.class);
      conf.setReducerClass(WC_Reducer.class);
      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);
      FileInputFormat.setInputPaths(conf,new Path(args[0]));
      FileOutputFormat.setOutputPath(conf,new Path(args[1]));
      JobClient.runJob(conf);
   }
}
```

//output



//dsbda 12

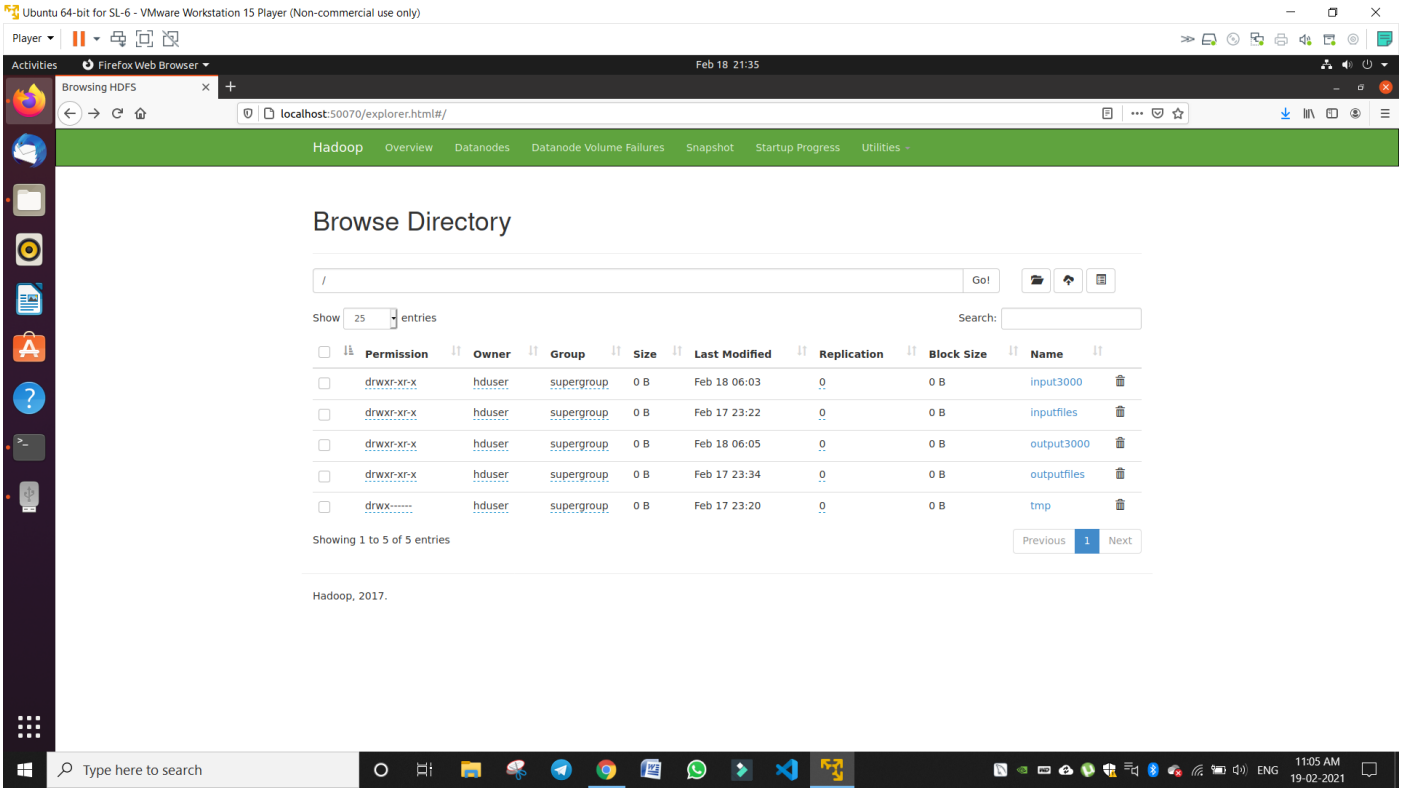
//SalesMapper.java

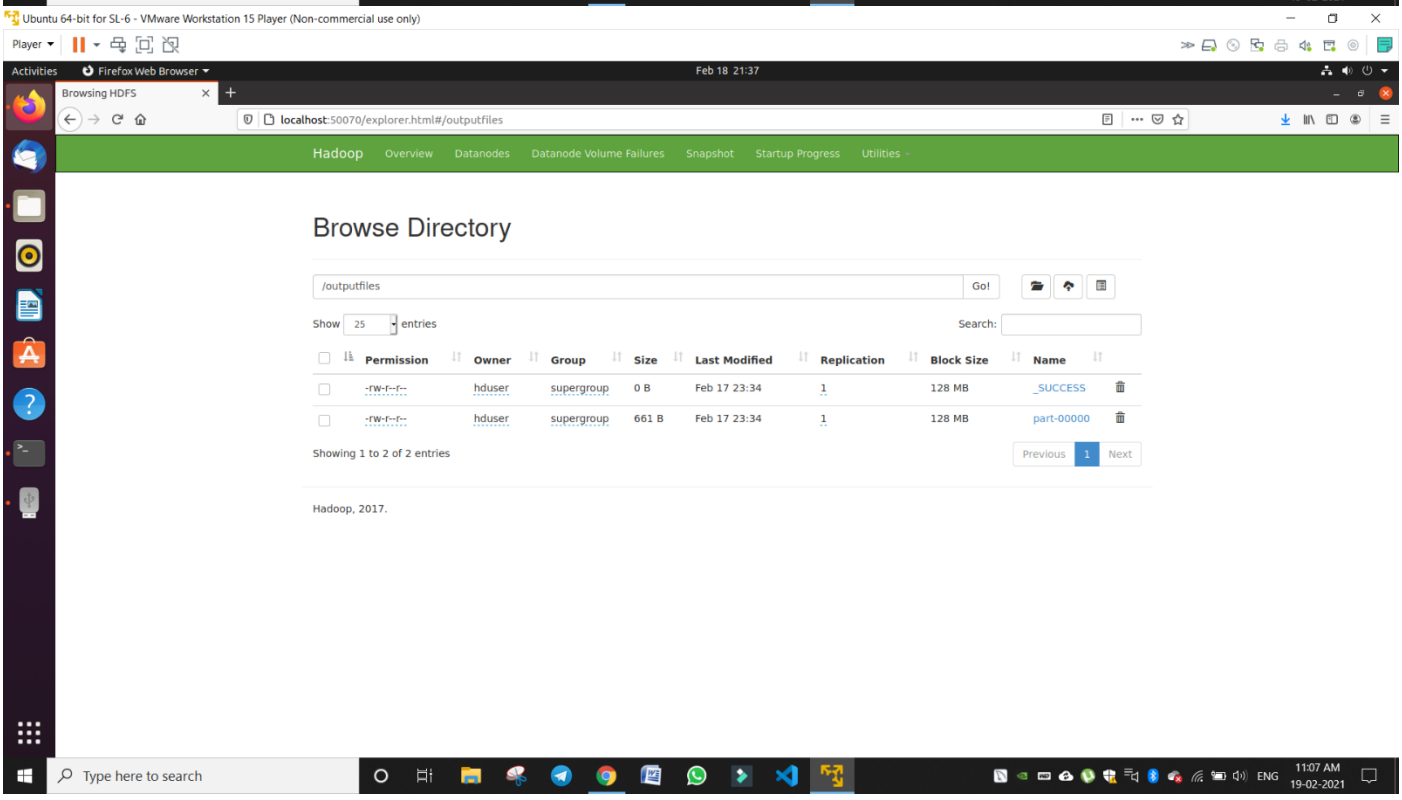
```
package SalesCountry;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class SalesMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text,
IntWritable> {
    private final static IntWritable one = new IntWritable(1);

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>
output, Reporter reporter) throws IOException {

        String valueString = value.toString();
        String[] SingleCountryData = valueString.split(",");
        output.collect(new Text(SingleCountryData[7]), one);
    }
}
```

//SalesCountryReducer.java

```
package SalesCountry;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class SalesCountryReducer extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {

    public void reduce(Text t_key, Iterator<IntWritable> values,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {
        Text key = t_key;
        int frequencyForCountry = 0;
        while (values.hasNext()) {
            // replace type of value with the actual type of our value
            IntWritable value = (IntWritable) values.next();
            frequencyForCountry += value.get();

        }
        output.collect(key, new IntWritable(frequencyForCountry));
    }
}
```

```
//SAlesCountryDriver.java

package SalesCountry;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class SalesCountryDriver {
    public static void main(String[] args) {
        JobClient my_client = new JobClient();
        // Create a configuration object for the job
        JobConf job_conf = new JobConf(SalesCountryDriver.class);

        // Set a name of the Job
        job_conf.setJobName("SalePerCountry");

        // Specify data type of output key and value
        job_conf.setOutputKeyClass(Text.class);
        job_conf.setOutputValueClass(IntWritable.class);

        // Specify names of Mapper and Reducer Class
        job_conf.setMapperClass(SalesCountry.SalesMapper.class);
        job_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);

        // Specify formats of the data type of Input and output
        job_conf.setInputFormat(TextInputFormat.class);
        job_conf.setOutputFormat(TextOutputFormat.class);

        // Set input and output directories using command line arguments,
        //arg[0] = name of input directory on HDFS, and arg[1] =  name of output directory
to be created to store the output file.

        FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

        my_client.setConf(job_conf);
        try {
            // Run the job
            JobClient.runJob(job_conf);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```


Ubuntu 64-bit for SL-6 - VMware Workstation 15 Player (Non-commercial use only)
Player
Activities
Firefox Web Browser
Feb 18 21:35
Browsing HDFS
localhost:50070/explorer.html#/
Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directory
/
Go!
Show 25 entries
Search:
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
drwxr-xr-x hduser supergroup 0 B Feb 18 06:03 0 0 B input3000
drwxr-xr-x hduser supergroup 0 B Feb 17 23:22 0 0 B inputfiles
drwxr-xr-x hduser supergroup 0 B Feb 18 06:05 0 0 B output3000
drwxr-xr-x hduser supergroup 0 B Feb 17 23:34 0 0 B outputfiles
drwx------ hduser supergroup 0 B Feb 17 23:20 0 0 B tmp
Showing 1 to 5 of 5 entries
Previous
1
Next
Hadoop, 2017.
Type here to search
ENG
11:05 AM
19-02-2021

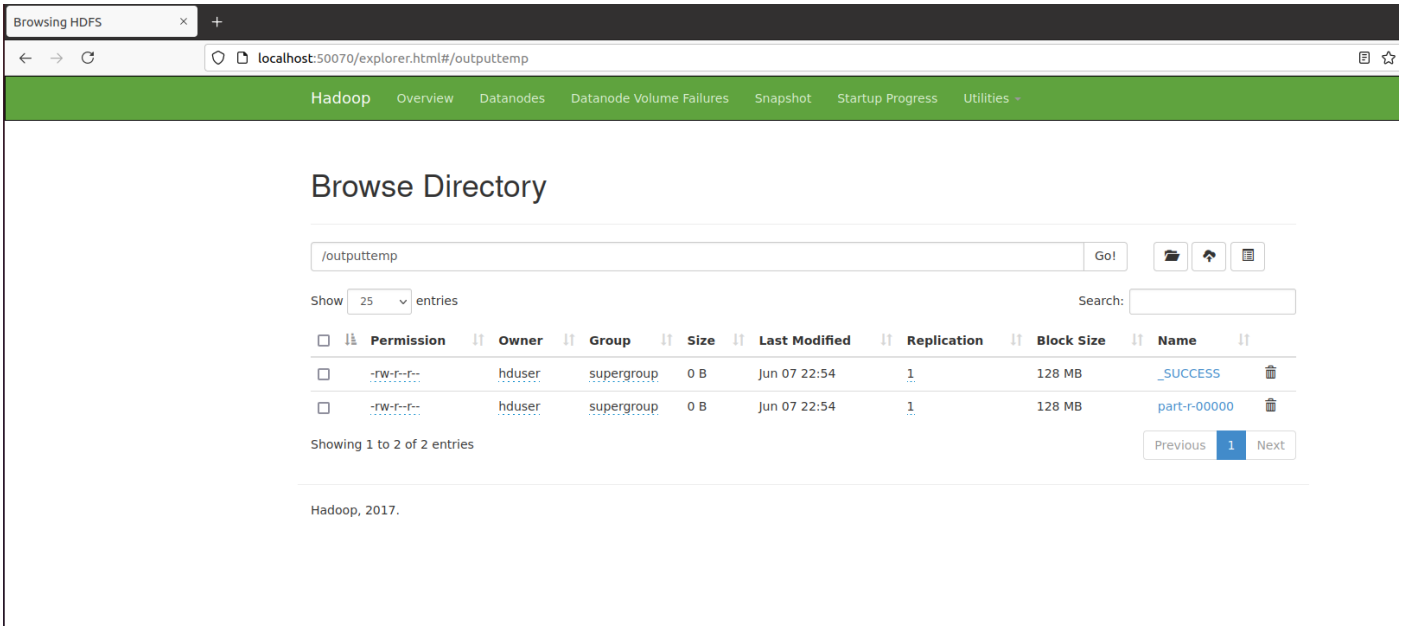

Ubuntu 64-bit for SL-6 - VMware Workstation 15 Player (Non-commercial use only)
Player
Activities
Firefox Web Browser
Feb 18 21:37
Browsing HDFS
localhost:50070/explorer.html#/outputfiles
Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directory
/outputfiles
Go!
Show 25 entries
Search:
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
-rw-r--r-- hduser supergroup 0 B Feb 17 23:34 1 128 MB _SUCCESS
-rw-r--r-- hduser supergroup 661 B Feb 17 23:34 1 128 MB part-00000
Showing 1 to 2 of 2 entries
Previous
1
Next
Hadoop, 2017.
Type here to search
ENG
11:07 AM
19-02-2021

//dsbda 13
//MaxTemperatureDriver.java

```
package MaxMinTemp;

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

/*This class is responsible for running map reduce job*/
public class MaxTemperatureDriver extends Configured implements Tool {
    public int run(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: MaxTemperatureDriver <input path> <outputpath>");
            System.exit(-1);
        }
        Job job = new Job();
        job.setJarByClass(MaxTemperatureDriver.class);
        job.setJobName("Max Temperature");
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setMapperClass(MaxTemperatureMapper.class);
        job.setReducerClass(MaxTemperatureReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
        boolean success = job.waitForCompletion(true);
        return success ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        MaxTemperatureDriver driver = new MaxTemperatureDriver();
        int exitCode = ToolRunner.run(driver, args);
        System.exit(exitCode);
    }
}
```

```
//MaxTemperatureReducer.java
package MaxMinTemp;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MaxTemperatureReducer
        extends Reducer<Text, IntWritable, Text, IntWritable> {
    @Override
    public void reduce(Text key, Iterable<IntWritable> values,
            Context context)
            throws IOException, InterruptedException {
        int maxValue = Integer.MIN_VALUE;
        for (IntWritable value : values) {
            maxValue = Math.max(maxValue, value.get());
        }
        context.write(key, new IntWritable(maxValue));
    }
}

//MaxTemperatureMapper.java
package MaxMinTemp;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MaxTemperatureMapper extends
        Mapper<LongWritable, Text, Text, IntWritable> {
    private static final int MISSING = 9999;

    @Override
    public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
        String line = value.toString();
        String year = line.substring(15, 19);
        int airTemperature;
        if (line.charAt(87) == '+') { // parseInt doesn't like leading plus
            // signs
            airTemperature = Integer.parseInt(line.substring(88, 92));
        } else {
            airTemperature = Integer.parseInt(line.substring(87, 92));
        }
        String quality = line.substring(92, 93);
        if (airTemperature != MISSING && quality.matches("[01459]")) {
            context.write(new Text(year), new IntWritable(airTemperature));
        }
    }
}
```

//output


Browsing HDFS
localhost:50070/explorer.html#/outputtemp
Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directory
/outputtemp
Go!
Show 25 entries
Search:
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
-rw-r--r--
hduser
supergroup
0 B
Jun 07 22:54
1
128 MB
_SUCCESS
-rw-r--r--
hduser
supergroup
0 B
Jun 07 22:54
1
128 MB
part-r-00000
Showing 1 to 2 of 2 entries
Previous
1
Next
Hadoop, 2017.